----------(C E E M A C)tm----------

THE VISUAL COMPOSITION SYSTEM

(RELEASE 1.1)

CEEMAC IS THE LANGUAGE OF FIRE ORGAN,
WHICH WAS THE 1ST 'SOFT VISUAL ALBUM'

A SYSTEM TO CREATE PERFORMANCE LEVEL
ABSTRACT VISUALS TO ACCOMPANY MUSIC
HAS ELUDED OUR GRASP FOR EONS. MY
PERSONAL SEARCH HAS LED TO CEEMAC.

JAN 1982 BROOKE W BOERING

DEDICATION

TO WASSILY KANDINSKY. FATHER OF ABSTRACT ART. VISIONARY WITHOUT PEER. SPIRITUAL MENTOR TO ME.

HE SAID -

...PURE MOTION. WHICH IS THE PRINCIPAL ELEMENT OF DANCE. APPLIES TO PAINTING. TOO... BOTH ARTS MUST LEARN FROM MUSIC THAT EVERY DISCORD THAT SPRINGS FROM INTERNAL NECESSITY IS BEAUTIFUL.

INTELLIGENCE. AWARENESS. LUCID INTENTION AND PRECISE AIM PLAY A PRIMARY PART; BUT IT IS NEVER CALCULATION WHICH TRIUMPHS. ALWAYS INTUITION.

...1912

THE PROGRESS WON THRU SYSTEMATIC WORK WILL CREATE AN ELEMENTARY DICTIONARY WHICH. IN ITS FURTHER DEVELOPMENT. WILL LEAD TO A 'GRAMMER' AND. FINALLY. TO A THEORY OF COMPOSITION WHICH WILL PASS BEYOND THE BOUNDRIES OF THE INDIVIDUAL ART EXPRESSIONS AND BECOME APPLICABLE TO 'ART' AS A WHOLE.

...1926

IT IS NOT OBVIOUS GEOMETRIC CONFIGURATIONS THAT WILL BE THE RICHEST IN POSSIBILITIES. BUT HIDDEN ONES. EMERGING UNNOTICED FROM THE CANVAS AND MEANT FOR THE SOUL RATHER THAN THE EYE.

THE FINAL ABSTRACT EXPRESSION OF EVERY ART IS NUMBER.

...1933

TABLE OF CONTENTS

INTRODUCTION

CEEMAC IS THE VISUAL COMPOSITION SYSTEM DESIGNED BY BRODKE W BOERING AND MADE AVAILABLE THRU VAGABONDO ENTERPRISES. IT CONTAINS TWO MAJOR COMPONENTS; AN INTERPRETER TO EXECUTE VISUAL SCORES AND AN EDITOR TO CREATE, MODIFY AND PRINT 'SCORES'. IT REQUIRES AN APPLE][COMPUTER RUNNING DOS 3.3 WITH AT LEAST 48K OF RAM MEMORY.

IF YOU CAN PROGRAM IN BASIC, YOU CAN HANDLE CEEMAC. EVEN IF YOU'VE NEVER PROGRAMMED IN A HIGH LEVEL LANGUAGE, CEEMAC SHOULD STILL BE RELATIVELY EASY TO PICK UP. HOWEVER, INSTEAD OF 'WRITING A PROGRAM', YOU 'COMPOSE A SCORE'.

VARIABLES ARE ALL PRENAMED AND PREDEFINED. MOST ARE SYSTEM SUPPORTED SUCH AS KEYBOARD AND PADDLE READINGS AND CONTINUALLY REFLECT CHANGING CONDITIONS. OTHERS PROVIDE RANGE-CONTROLLED VALUES FOR RANDOM, FIXED-RANDOM AND TABLE DRIVEN NEEDS.

A POWERFUL RESIDENT MACRO LIBRARY IS DIRECTLY CALLABLE FROM SCORES. MANY DRAWING FUNCTIONS AND COMMONLY REQUIRED MANIPULATIONS ARE PERFORMED WITH LITTLE NEED FOR CUSTOM PROGRAMMING. THESE FEATURES ARE DESIGNED TO ASSIST THE VISUAL COMPOSER IN CREATING COMPOSITIONS THAT ARE EFFICIENT IN EXECUTION AND PLEASANT TO VIEW.

THE APPLE-SUPPORTED 3-BIT FORMAT SHAPE SYSTEM WAS FOUND TO BE INTOLERABLY LIMITED. IT SIMPLY COULDN'T SUPPORT THE THINGS THAT NEEDED DOING AND A NEW SHAPE SYSTEM HAS BEEN DESIGNED FOR CEEMAC. AS OF RELEASE 1.0, SHAPE EXECUTION IS FULLY SUPPORTED BUT SHAPE GENERATION IS NOT. A DETAILED DESCRIPTION OF THE SYSTEM STRUCTURE IS CONTAINED IN A SEPARATE CHAPTER WHICH SHOULD ALLOW THOSE WITH SUFFICIENT INTEREST TO CREATE THEIR OWN SHAPES AND SHAPE TABLES.

THE CONCEPTS AND USAGE OF 'LISTS' IN CEEMAC IS ALSO EXPLAINED IN SOME DETAIL. MEMORY FOR 4 SUCH LISTS HAS BEEN SET ASIDE AND COMPOSERS CAN CREATE THEIR OWN LIBRARIES WITH ONLY A LITTLE KNOWLEDGE OF BINARY FILE SUPPORT UNDER DOS.

ONE LEARNS EARLY TO NEVER START A COMPUTER PROGRAM (OR SCORE) FROM SCRATCH. BEGIN WITH SOMETHING YOU'VE DONE BEFORE THAT'S SORTA CLOSE TO WHAT YOU WANT TO DO. THEN START CHANGING THINGS. SINCE YOU HAVEN'T WRITTEN A SCORE YET, SIMPLY START WITH ONE OF THE SAMPLES THAT ACCOMPANY THIS SYSTEM. PICK ONE THAT YOU LIKE OR WOULD LIKE TO EXPAND UPON.

AN UNDERSTANDING OF TERMINOLOGY USED WITH THE BASIC LANGUAGE CAN HELP WITH THE JARGON BUT NEW TERMS AND CONCEPTS ARE INTRODUCED AS WELL. WHENEVER THE MANUAL GETS TO BE HARD READING, STOP AND EXPERIMENT WITH A SCORE. I CANT EMPHASIZE ENOUGH HOW IMPORTANT THIS 'LEARNING BY DOING' PROCESS IS TO SUCCESSFUL VISUAL COMPOSITION. YOU'LL FIND CEEMAC TO BE 'FOOLING-AROUND-PROOF'.

CEEMAC SUPPORTS DECIMAL VALUES RELUCTANTLY. IF YOU ARE UNFAMILIAR WITH HEX NOTATION, THIS IS YOUR BEST CHANCE TO LEARN. VISUALIZING SCREEN RELATIONSHIPS CAN BE FAR EASIER WITH HEX THAN DECIMAL. IN HEX, THERE ARE 16 POSSIBLE VALUES BETWEEN 0 AND 'F'. THESE DIGITS ARE SOMETIMES CALLED 'NIBBLES' AND TWO COMBINE TO FORM A BYTE. CONVENTIONALLY, HEX-NOTATED BYTES ARE PRECEEDED BY A '$' SIGN (E.G. $B9). IN THIS 8-BIT WORLD $FF IS KING AND VALUES SIMPLY DONT EXCEED 255 (DECIMAL). IF THIS IS ALL NEWS TO YOU, TRY WORKING WITH HEX FOR A WHILE AND IF YOU FIND IT TOO UNCOMFORTABLE, SIMPLY USE THE DECIMAL SUPPORT PROVIDED.

THERE ARE THREE 'MODES' IN THIS SYSTEM. THEY ARE DOS, EXECUTION AND EDITING. BOOT-UP STARTS THE TITLE SCREEN (SLIGHTLY DIFFERENT THAN THE FIRE ORGAN ALBUM). WHEN THE COVER IMAGE STARTS CHANGING, YOU'RE IN EXECUTION MODE JUST AS YOU MIGHT BE WHEN RUNNING A BASIC PROGRAM. FROM HERE YOU CAN GET TO EDIT MODE WITH A CTRL-A OR TO DOS WITH A CTRL-C.

ONCE IN EDIT MODE, THE SCORE CAN BE MODIFIED AS EXPLAINED IN THE FOLLOWING CHAPTERS. FROM THERE YOU CAN RETURN TO EXECUTION MODE WITH A CTRL-A OR EXIT TO DOS WITH A CTRL-C.

WHEN IN DOS, YOU MAY DO ALL THOSE THINGS YOU WOULD EXPECT (EXCEPT DONT WRITE OR RUN OTHER PROGRAMS). YOU MAY, HOWEVER, BRUN A SCORE, LOAD A NEW

SHAPE TABLE OR A DIFFERENT 'LISTS' MODULE. TO RETURN TO EXECUTION MODE, DO A 'CALL 2048' (OR *800G). FROM THERE YOU CAN GET BACK TO EDITING BY KEYING A CTRL-A.

'DOS TRANSPARENCY' IS WHEN A SYSTEM 'HIDES' THE RIGOROUS PROTOCOL OF DOS. AT PRESENT, DOS TRANSPARANCY DOES NOT EXIST IN CEEMAC. HOWEVER, EXITING TO DOS FROM EDIT MODE WILL BUILD THREE 'BSAVE' LINES FOR YOUR POSSIBLE USE. THIS 'EXIT HANDLER' ASSUMES THAT YOU (MIGHT) WANT TO NAME THE SCORE FILE SAME AS THE TITLE LINE AND IMBEDS THAT INTO THE 'BSAVE' LINE. ALTER THIS FILE NAME AS YOU WISH. ONE CAUTION, LONG SCORE NAMES MAY CAUSE THE BSAVE LINE TO WRAP AROUND TO THE START OF A SECOND LINE. SINCE THE SCORE LENGTH IS AT THE END, IT IS BEST TO MAKE IT A HABIT TO ALWAYS SCROLL A FEW BLANKS PAST YOUR LINE END WHEN DOING THE BSAVE. YOU'LL ONLY HAVE TO TRUNCATE A PRIZE SCORE ONCE TO SEE THE WISDOM IN THIS. SCORES SAVED THIS WAY CAN BE 'BRUN' FROM DOS MODE ANYTIME AFTER CEEMAC HAS BEEN BOOTED.

YOU CAN ALSO BSAVE THE SHAPES TABLE AND/OR THE 'LISTS' MODULE. THE LISTS MODULE STARTS AT $5000 AND IS USUALLY 4 PAGES LONG ($400). THE SHAPE TABLE STARTS AT $4400 AND CAN HAVE A LENGTH AS LONG AS $C00. AT THIS TIME, LOCATION OF INDIVIDUAL SHAPES IN THE TABLE CAN ONLY BE DETERMINED BY DETAILED EXAMINATION (AS WAS TRUE WHEN THE APPLE II ORIGINALLY APPEARED).

DURING EXECUTION, THE KEYBOARD SERVES AS A COMMAND INPUT DEVICE. DETAILS ON THIS ARE PROVIDED IN THE NEXT CHAPTER. IN ADDITION TO BEING ABLE TO CHANGE MODES (AS ABOVE), YOU WILL BE ABLE TO AFFECT SCORE EXECUTION IN MANY WAYS DEPENDING MAINLY UPON HOW THE SCORE IS WRITTEN. THE PADDLES AND BUTTONS AUGMENT THIS CAPABILITY.

REFERENCES TO A PARTICULAR 'BIT-IN-BYTE' HAS BEEN ONE OF THE GREAT 'NON STANDARDS' OF COMPUTERDOM. I KNOW OF AT LEAST 4 'STANDARDS' FOR THIS AND NONE OF THEM EVER MADE ANY SENSE. AT THE RISK OF ADDING TO THE CONFUSION, I'VE ADOPTED THE FOLLOWING FOR CEEMAC:

```
  BIT01 MEANS: 0000 0001
  BIT02 MEANS: 0000 0010
  BIT08 MEANS: 0000 1000
 *BIT30 MEANS: 0011 0000
 *BIT77 MEANS: 0111 0111  ETC, ETC.
```

*INDICATES MULTIPLE BITS 'OR'ED.

I'LL NOT ATTEMPT TO JUSTIFY OR RATIONALIZE IT FURTHER; ITS JUST MY TERMINOLOGY.

EXECUTION COMMANDS

SPECIAL NOTE: THIS CHAPTER CAN BE MOST CONFUSING FOR THE NEW VISUAL COMPOSER TRYING TO GET A 'HANDLE' ON CEEMAC. SOME OF THESE COMMANDS ARE QUITE OBSCURE (LIKE THE 'DRUM KEYS') WHILE OTHERS ARE USED FREQUENTLY AND SHOULD BE LEARNED HERE. I SUGGEST THAT YOU IGNORE (FOR PRESENT) ANYTHING THAT IS STILL CONFUSING WHEN READ FOR THE 2ND TIME. YOU CAN ALWAYS RETURN TO THIS LATER WHEN YOU'RE MORE FAMILIAR WITH CEEMAC.

KEYS 1-5:

EACH TIME ONE OF THESE KEYS IS HIT, ITS CORRESPONDING 'DRUM' (DRUM1, DRUM2, ETC) IS FLIPPED FROM ON (0) TO OFF (NON-ZERO) OR VISA VERSA. IN OTHER WORDS THEY ACT LIKE SWITCHES. THE VARIABLES DRUM1 THRU DRUM5 REFLECT THESE CHANGES. THE COMPOSER CAN INCLUDE TESTS OF THESE VARIABLES TO HAVE THE SCORE RESPOND DYNAMICLY TO EXECUTION TIME PERFORMANCE. (SEE 'VARIABLES' AND 'MUSIC' CHAPTERS).

SPACE BAR:

NORMALLY DEDICATED TO PICTURE FREEZE, THIS KEY HAS NO OTHER USE. IT CAN BE RELEASED FROM THIS PURPOSE BY THE PROCEDURE DESCRIBED AFTER NEXT.

COLON:

NORMALLY UNUSED, THIS KEY CAN BE ASSIGNED TO PICTURE FREEZE BY THE PROCEDURE DESCRIBED NEXT.

ASTERISK (*):

THIS IS AN 'EXECUTION SWITCH'. ITS SOLE PURPOSE IS TO CHANGE WHICH KEY IS ASSIGNED TO THE FUNCTION OF PICTURE FREEZE (SEE ABOVE). THE ANTICIPATED USE IS WHEN THE KEYBOARD 'PERFORMER' PLAYING THE 'DRUMS' FINDS HIMSELF UNINTENIONALLY FREEZING THE PICTURE WITH THE HEAL OF HIS LEFT HAND. BY THROWING THIS SWITCH, THE FREEZE FUNCTION GETS MOVED SAFELY OUT OF RANGE (TO THE COLON KEY). HITTING THIS KEY AGAIN RETURNS THE FREEZE FUNCTION TO THE SPACE BAR.

SPECIAL NOTE:

LOOKING BACK OVER THE KEYS COVERED SO FAR WILL INDICATE HOW THEY TEND TO WORK TOGETHER DURING DRUM PLAY. THE PICTURE FREEZE FUNCTION IS, OF COURSE, PRESENT WHETHER PLAYING DRUMS OR NOT.

ALPHABETIC KEYS (A-Z):

THE ORIGINAL FIRE ORGAN ALBUM CONTAINED 35 SCORES (AND NO EDITOR). EACH OF THOSE SCORES HAD TO BE 64 STATEMENTS OR LESS. RELEASE 1.0 OF CEEMAC PERMITS SCORES AS LONG AS 256 STATEMENTS, CONTAINS THE FULL EDITOR AND HAS ADDED FEATURES. AS A RESULT, MULITPLE SCORES CANNOT NORMALLY CO-EXIST AS THEY DID ON THE FIRE ORGAN ALBUM.

FROM THE VIEW OF THE INTERPRETER, THE ALPHABETIC KEYS ARE ALL INVALID. WHENEVER AN INVALID KEY IS HIT, TWO THINGS HAPPEN. FIRST, A SOFT 'CLICK' IS SOUNDED THRU THE APPLE SPEAKER. SECOND, THE ASCII VALUE (HI-BIT ON) OF THE INVALID KEYIN IS PLACED IN A SCORE-READABLE VARIABLE NAMED 'KEY'.

WHILE ONLY ONE (LARGER) SCORE CAN BE IN MEMORY AT ONE TIME, THAT SCORE CAN BE DESIGNED TO RESPOND TO THE ALPHABETIC KEYS. BY TESTING THE 'KEY' VARIABLE FOR SPECIFIC KEYS (OR NON-ZERO), THE SCORE CAN 'KNOW' WHEN ONE OF THESE KEYS HAS BEEN HIT AND TAKE A DIFFERENT EXECUTION COURSE.

COMPOSERS WHO CHOOSE TO UTILIZE THIS FEATURE SHOULD ALSO INCLUDE AN INSTRUCTION TO 'UNLOAD' THE 'KEY' VARIABLE BY RESETING IT TO ZERO AFTER NON-ZERO DETECTION. ANOTHER HIGHLY DESIRABLE ACTION WOULD BE TO INCLUDE A

STATEMENT SETTING BIT20 IN THE 'XOPT' VARIABLE (SEE VARIABLES). THIS WILL PREVENT THE (PROBABLY ANNOYING) AUDIBLE 'CLICK' WHEN THE INVALID KEY IS STRUCK. (ALSO, SEE NEXT)

EQUAL SIGN (=):

THIS KEY SERVES AS A 'NO CLICK ON COMMAND ERROR' SWITCH. IT IS THE DYNAMIC EQUIVALENT TO SETTING BIT20 IN THE 'XOPT' VARIABLE. WHILE NORMALLY OFF (0), IT CAN BE FREELY SWITCHED AS THE OCCASION DEMANDS. (SEE ABOVE)

RETURN KEY:

THIS KEY IS USED TO 'REVERSE' THINGS ON THE SCREEN PROVIDING THE SCORE HAS ANTICIPATED ITS USE. A COMPLETE DESCRIPTION CAN BE FOUND IN THE CHAPTER ON 'REVERSING'.

'(--' KEY:

THIS KEYIN CAUSES THE INTERPRETER TO 'GOTO 0'. IF THE SCORE HAS NO SYMBOL-0, THE REVRSW WILL BE RESET WITH THE VALUE IN THE REVCNT VARIABLE (SEE 'REVERSING' CHAPTER). BUTTON-0 (ON PADDLE-0) ACTS AS A REMOTE EQUIVALENT TO THIS KEY.

'--)' KEY:

THIS KEYIN CAUSES THE INTERPRETER TO 'GOTO 1'. IF THE SCORE HAS NO SYMBOL-1, EXECUTION WILL RESTART FROM THE TOP BUT DOES NOT REINITIALIZE VARIABLES. BUTTON-1 (ON PADDLE-1) ACTS AS A REMOTE EQUIVALENT TO THIS KEY.

'J' KEY:

THIS IS A 1-WAY SWITCH THAT SIGNALS THE INTERPRETER TO START CONTINUOUS READING OF THE CASSETTE IN PORT FOR MUSIC INPUT (SEE 'MUSIC' CHAPTER). THIS ACTION CAN ONLY BE REVERSED BY KEYING A CARRET (^) (SEE BELOW).

'^' KEY:

THIS TURNS OFF READING OF THE CASSETTE-IN PORT.

GREATER-THAN ()):

THIS IS A SWITCH THAT OPENS AND CLOSES THE 'TRACE' WINDOW (SEE 'MACROS' CHAPTER).

CTRL-A:

THIS KEY HALTS EXECUTION AND TRANSFERS CONTROL TO THE EDITOR.

CTRL-C:

THIS KEY HALTS EXECUTION AND TRANSFERS CONTROL TO DOS.

CTRL-SHIFT-L:

THIS DIFFICULT-TO-DO KEY INPUT WILL VIRTUALLY LOCK THE KEYBOARD. OTHER THAN THE 'RESET' KEY, THE FREEZE KEY OR A REPEAT OF THIS COMMAND ITSELF, ALL KEYINS ARE CONSIDERED INVALID. THE AUDIBLE 'CLICK' FOR INVALID KEYINS WILL NOW BE A SOMEWHAT LOUDER, RASPIER SHORT BUZZ. THIS CAN SERVE AS A REMINDER THAT THIS 'LOCK' HAS BEEN THROWN.

WHY WOULD ANYONE WANT TO LOCK THE KEYBOARD IN THIS MANNER? SOMEONE ONCE DEFINED A 'BULLET' AS A TEN YEAR OLD KID THAT HITS EVERY KEY ON THE KEYBOARD

EVERY CHANCE HE GETS. I THINK THAT'S GREAT!!.. EXCEPT WHEN I WANT TO LEAVE A SCORE RUNNING WITHOUT KEYBOARD INPUTS. YOU MIGHT WANT TO 'BULLETPROOF' A PERFORMANCE THIS WAY AT TIMES.

BUTTON-0:

SEE '(--' KEY ABOVE. ALSO, WORKS 'BUT0' VARIABLE LIKE A SWITCH.

BUTTON-1:

SEE '--)' KEY ABOVE. ALSO, WORKS 'BUT1' VARIABLE LIKE A SWITCH.

BUTTON-2:

WORKS 'BUT2' VARIABLE LIKE A SWITCH. IF THIS BUTTON IS UNAVAILABLE, THE READING WILL BE $80.

PADDLE-0 (PDL0):

NORMALLY ASSIGNED AS A CRUDE SPEED CONTROL DURING EXECUTION. THIS FUNCTION CAN BE DEACTIVATED BY THE INCLUSION OF THE 'SPEED' MACRO IN THE SCORE (SEE 'MACROS' CHAPTER). ALSO, THE CURRENT READING (0-$FF) CAN BE FOUND IN 'PDL0' VARIABLE.

PADDLE-1 (PDL1):

ALWAYS SUPPORTS THE 'EXTRACTED' VARIABLES GROUP (SEE 'VARIABLES' CHAPTER). ALSO, THE CURRENT READING (0-$FF) CAN BE FOUND IN 'PDL1' VARIABLE.

PADDLE-2 (PDL2):

PUTS THE CURRENT READING (0-$FF) INTO 'PDL2' VARIABLE. IF THIS PADDLE IS UNAVAILABLE, THE READING WILL BE $FF.

PADDLE-3 (PDL3):

PUTS THE CURRENT READING (0-$FF) INTO 'PDL3' VARIABLE. IF THIS PADDLE IS UNAVAILABLE, THE READING WILL BE $FF.

SPECIAL NOTE: IN THE EARLY DAYS, ALL APPLE II COMPUTERS CAME WITH A SET OF PADDLES (PDL0 & PDL1). ALMOST NO ONE THESE DAYS HAS PADDLES. IF YOU HAVE A JOYSTICK, PDL0 IS THE X-AXIS READING, PDL1 IS THE Y-AXIS. IF YOU HAVE A 2ND JOYSTICK, PDL2 IS ITS X-AXIS AND PDL3 IS ITS Y-AXIS. IF YOU HAVE NEITHER PADDLES NOR JOYSTICKS, JUST IGNORE THESE FUNCTIONS.

FROM EXECUTION MODE, A CTRL-A WILL CAUSE ENTRY INTO 'EDIT' MODE. HERE THE CURRENT SCORE WILL BE DISPLAYED WITH THE ARROW CURSOR POINTING TO THE LAST EDITED LINE. EACH LINE IS A 'STATEMENT' OR COMMENT LINE WITH INDENTATION CORRESPONDING TO ITS RELATIVE 'DEPTH' IN A LOOP OR 'SUB'. THIS CHAPTER DISCUSES THE VERTICAL SCROLLING PROCEDURE AND DESCRIBES THE CONTROL STATEMENTS IN DETAIL.

THE FOLLOWING COMMANDS APPLY FOR THE BLINKING ARROW CURSOR AT THE LEFT MARGIN:

```
  SCROLL UP = UP-ARROW (OR '-' KEY)
  SCROLL DOWN = DOWN-ARROW (OR RETURN)
  LIST FROM TOP = 'L'
  DELETE A STATEMENT = 'D'
 *REPLACE A STATEMENT = 'R'
 *INSERT A STATEMENT = 'I'
 *RE-EDIT A STATEMENT = '-->'
  OVERDRIVE SWITCH = 'O'
  RETURN TO EXECUTION = CTRL-A
  EXIT TO DOS MODE = CTRL-C
```

*WHEN YOU DO AN 'R, I OR -->' THE ARROW CURSOR FREEZES AND YOU ARE NOW IN HORIZONTAL SCROLLING MODE WITH THE FAMILIAR BLINKING BLOCK CURSOR AT THE FIRST CHARACTER OF THE LINE. RETURN TO VERTICAL SCROLLING BY DOING A '<--'. EDITING STATEMENT LINES IS EXPLAINED IN THE NEXT CHAPTER.

RAPID SCROLLING IS ACCOMODATED BY HOLDING THE 'REPT' KEY DOWN ALONG WITH THE DESIRED SCROLL KEY. THE OVERDRIVE SWITCH PERMITS FASTER VERTICAL SCROLLING. IT CHANGES THE APPEARANCE OF THE ARROW CURSOR SLIGHTLY AND IS TURNED OFF WHENEVER SCROLLING DIRECTION IS REVERSED.

NESTED LOOPS AND 'SUB'S ARE 2-SPACE INDENTED TO AIDE VISUALIZATION.

EACH SCORE IS LIMITED TO A MAXIMUM OF 256 STATEMENTS. THE FIRST STATEMENT IS THE TITLE LINE AND STARTS WITH 'SCORE:'. THE LAST STATEMENT SAYS 'CEEMAC' AND SHOWS THE RELEASE#. IT IS CALLED THE 'FENCE' STATEMENT. IN ADDITION TO THESE (WHICH CEEMAC ENFORCES) THERE ARE EIGHT 'EXECUTION CONTROL' STATEMENTS. A 'COMMENTS' STATEMENT AND A 'SYM'BOL STATEMENT MAKING A TOTAL OF 12. HERE THEY ARE:

```
      SCORE:   (N-CHAR TITLE LINE)

      :        (N-CHAR COMMENT LINE)

      (SYM)    (N-CHAR COMMENT LINE)
      GOTO     (SYM) (C-EXPRSN)

      GOSUB    (SUB IDENT; 1ST 3 CHARS)
      SUB      (N-CHAR IDENTIFIER)

      DO       (C-EXPRSN)
      AGAIN    (C-EXPRSN)

      FOR      (M-ESPRSN)

      SKIP     (C-EXPRSN)
      EXIT     (C-EXPRSN)

      CEEMAC  REL 1.1
```

IF SOMETHING IS WITHIN NORMAL PARENTHISIS '()', IT MUST BE PRESENT. SOMETHING WITHIN '()' MEANS THAT IT IS OPTIONAL.

'N-CHAR' LENGTHS ARE LIMITED TO THE AVAILABLE LINE SPACE. THIS WILL VARY DEPENDING UPON CURRENT INDENTATION.

(SYM) MEANS THAT A SINGLE HEX DIGIT (0 THRU F) MUST BE PRESENT, NOT THE ACTUAL WORD 'SYM'.

C-EXPRSN = CONDITIONAL EXPRESSION.

M-EXPRSN = MATH-A-LOGICAL EXPRESSION.

(BOTH OF THE ABOVE ARE EXPLAINED IN DETAIL A LITTLE LATER).

SCORE: (N-CHAR TITLE LINE)

DRESS UP YOUR SCORE WITH A NEATLY CENTERED TITLE LINE. AVOID USING IMBEDDED COMMAS HERE BECAUSE THEY CANT BE INCLUDED IN FILE NAMES. WHEN YOU START TO ALTER A SCORE (THE NORMAL WAY TO CREATE NEW SCORES), IT IS SUGGESTED THAT YOU REEDIT THE TITLE LINE FIRST TO A UNIQUE STRING. THIS SHOULD LESSEN THE RISK OF OVER-BSAVING LATER TO THE STARTING SCORE WITH THE OLD NAME. BE SURE TO START THE SCORE NAME WITH AN ALPHABETIC CHARACTER (A-Z) TO APPEASE DOS FILENAME STANDARDS.

: (N-CHAR COMMENT LINE)

USE AS MANY OR AS FEW CHARACTERS AS YOU WISH TO, MAKING YOUR OWN TRADEOFFS AGAINST STATEMENT SPACE. EACH 4 CHARACTERS (IN EXCESS OF THE FIRST 3) REDUCE THE AVAILABLE STATEMENT COUNT BY ONE SINCE THEY SHARE THE SAME MEMORY SPACE.

(SYM) (N-CHAR COMMENT LINE)

ONLY THE 16 HEX-DIGITS (0 THRU F) ARE AVAILABLE FOR SYMBOL REFERENCES. THESE REFERENCES ARE USEFUL PRIMARILY FOR 'GOTO' STATEMENTS. COMMENTS MAY BE APPENDED OR NOT, AS YOU SEE FIT.

SYM-0, 1 & F ARE NOT PERMITTED IN LOOPS OR SUBS.

SYM-0; SEE EXPLANATION IN 'EXECUTION CMDS' CHAPTER UNDER THE '(--' KEY.

SYM-1; SEE EXPLANATION IN 'EXECUTION CMDS' CHAPTER UNDER THE '-->' KEY.

SYM-F IS THE RESTART POINT WHENEVER THE END-OF-MAIN-CHAIN IS ENCOUNTERED (AT THE FIRST 'SUB' STATEMENT OR AT THE 'CEEMAC' (FENCE) STATEMENT).

GOTO (SYM) (C-EXPRSN)

THIS IS A JUMP TO THE GIVEN SYMBOL. IF NO (C-EXPRESN) IS PROVIDED THEN THE JUMP IS UNCONDITIONAL.

GOSUB (SUB IDENTIFIER: 1ST 3 CHARS ONLY)

THIS IS A CALL TO THE SPECIFIED SUBROUTINE. IT WORKS LIKE YOU WOULD EXPECT, RETURNING TO THE STATEMENT FOLLOWING THE GOSUB AFTER EXECUTION. PRESENTLY, THERE IS NO DIRECT WAY TO PASS PARAMETERS EXCEPT FOR THE CALLER TO HAVE SET UP 1 OR MORE VARIABLES WHICH THE SUB WILL 'KNOW' ARE FOR ITS USE.

ONLY THE FIRST 3 CHARACTERS OF A CALLED SUB MAY BE USED HERE. THIS IS A DESIGN TRADEOFF. IN THE FUTURE THIS MIGHT ACTUALLY PROVE TO BE A CONVENIENCE WHEN WORKING WITH SUB-LIBRARIES.

SUB (N-CHAR IDENTIFIER)

THIS IDENTIFIES THE START OF A SUBROUTINE. ALL SUBROUTINES *MUST BE LOCATED* AT THE END OF THE SCORE.

IT IS RECOMMENDED THAT SELECTION OF THE FIRST 3 CHARS OF THE IDENTIFIER BE GIVEN SOME SPECIAL THOUGHT. PERHAPS NUMBERING YOUR 'SUB'S STARTING AT A01 OR B11 WOULD WORK WELL. SUCH A STATEMENT MIGHT LOOK LIKE THIS:

-->SUB A01 -DRAW/ERASE SHAPE#5-

DD (C-EXPRSN)

THIS IS THE NORMAL START-OF-LOOP STATEMENT. IT MUST HAVE A MATCHING 'AGAIN' STATEMENT. IF THE OPTIONAL (C-EXPRSN) IS OMITTED, THE LOOP CAN NEVER EXIT AT THIS STATEMENT.

AGAIN (C-EXPRSN)

THIS IS THE NORMAL END-OF-LOOP STATEMENT. IT MUST HAVE A MATCHING 'DO' STATEMENT. IF THE OPTIONAL (C-EXPRSN) IS OMITTED, THE LOOP CAN NEVER EXIT AT THIS STATEMENT.

NOTE: IT CAN BE SEEN THAT IF NEITHER THE MATCHED 'DO' OR 'AGAIN' STATEMENT CONTAINS A (C-EXPRSN), THE LOOP MIGHT CONTINUE INDEFINITELY. IF BOTH HAVE A (C-EXPRSN), EXIT CAN OCCUR IF EITHER CONDITION IS TRUE AT THE TIME IT IS EVALUATED.

FOR (M-EXPRSN)

THIS SERVES THE SAME PURPOSE AS THE 'DO' STATEMENT EXCEPT THAT IT WORKS MUCH LIKE THE BASIC 'FOR' STATEMENT. (M-EXPRSN) IS AN ARTHMETIC-LIKE EXPRESSION THAT IS REQUIRED (SEE 'MATH-A-LOGICAL' CHAPTER). IT IS EVALUATED AND 'FIXED' UPON ENTRY INTO THE LOOP. UNLIKE BASIC, IT REMAINS FIXED EVEN IF CHANGES OCCUR TO ANY VARIABLES USED. IF THE M-EXPRSN EQUALS ZERO, IT WILL EXECUTE ONCE.

SKIP (C-EXPRSN)

THIS STATEMENT WILL CAUSE THE NEXT STATEMENT TO BE NOT EXECUTED. IF THE (C-EXPRSN) IS PRESENT THEN THE COMMAND IS CONDITIONAL. NOTE: MAY NOT BE USED IMMEDIATELY PRIOR TO THE CONTROL STATEMENTS 'DO', 'FOR', 'AGAIN' OR 'SUB' NOR AT THE END OF THE MAIN CHAIN (THE STRING OF STATEMENTS BEFORE THE FIRST 'SUB' STATMENT). COMMENT LINES AND 'SYM' STATEMENTS DONT COUNT AS 'SKIP'ED STATEMENTS.

EXIT (C-EXPRSN)

THIS STATEMENT IS AN 'ESCAPE HATCH' WHICH CAN BE USED WITHIN LOOPS, SUBS OR THE MAIN CHAIN. IF ENCOUNTERED IN A LOOP, THE LOOP IS EXITED. IF FOUND IN A SUBROUTINE, THE SUB IS EXITED. DETECTED IN THE MAIN CHAIN, THE SCORE RESTARTS FROM THE TOP. IF THE (C-EXPRSN) IS PRESENT THE COMMAND IS CONDITIONAL.

CEEMAC REL 1.0

THIS IS THE FENCE STATEMENT. IT SERVES MANY PURPOSES, MOST OF THEM INTERNAL TO THE SYSTEM. IT CANNOT BE REMOVED OR MODIFIED IN ANY WAY. IF BY SOME CHANCE IT IS TRUNCATED (SUCH AS WHEN 'BSAVING') OR MANGLED THE INTERPRETER IS HELPLESS AND WILL CAUSE A SYSTEM ERROR BREAK. MORE IMPORTANTLY, THE SCORE IS PROBABLY BEYOND SAVING.

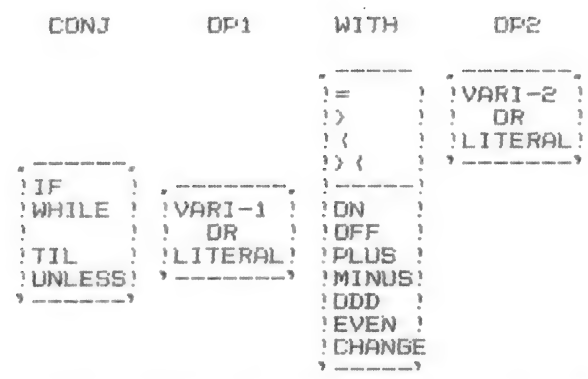
EXPRESSIONS

THESE OPERATE IN A QUITE NATURAL MANNER. THEIR TECHNICAL EXPLANATION, HOWEVER, CAN BE VERY HARD READING FOR THE UNINITIATED. ALSO, TERMINOLOGY MAY CAUSE SOME UNINTENDED CONFUSION. DONT BE AFRAID TO SKIP THIS IF YOU WISH. THE EXAMPLE SCORES WILL BE THE BEST TEACHER ANYWAY.

C-EXPRSN:

THIS MEANS 'CONDITIONAL EXPRESSION' AND IS USED ONLY WITH THE CONTROL STATEMENTS. ITS PURPOSE IS TO PERMIT EXECUTION TIME CONDITIONS TO DETERMINE STATEMENT EXECUTION PATHS. THE TECHNICAL EXPLANATION MAY BE COMPLEX BUT THE APPLICATION IS STRIGHTFORWARD.

C-EXPRSNS ARE OF THE FOLLOWING FORMAT:



NOTE THAT THE ALPHABETIC 'WITH' GUYS (WORDS) DO NOT REQUIRE OP2. IN THESE CASES, OP1 IS TESTED AS FOLLOWS:

WITH	TRUE IF:
ON	OP1 IS NOT ZERO
OFF	OP1 IS ZERO
PLUS	OP1 IN $0-$7F RANGE
MINUS	OP1 IN $80-$FF RANGE
EVEN	OP1 IS 0,2,4...$FC,$FE
ODD	OP1 IS 1,3,5...$FD,$FF
CHANGE	OP1 IS DIFFERENT THAN LAST

THE COMBINATION OF OP1/WITH/OP2 IS FIRST EVALUATED FOR TRUE/FALSE BY THE INTERPRETER. THE RESULT IS THEN MATCHED AGAINST THE CONJUNCTIVE WORD AS THE FOLLOWING TRUTH TABLE SHOWS:

CONJUNCTIVE	TRUE	FALSE
IF/WHILE	Y	N
TIL/UNLESS	N	Y

A 'Y' WILL CAUSE THE CONTROL STATEMENT TO EXECUTE, WHERE AN 'N' WILL PROHIBIT ITS EXECUTION.

M-EXPRSN:

THIS MEANS 'MATH-A-LOGICAL EXPRESSION' AND IS OF THE FORM:

OP1 (^ OP2)

NOTE THAT THE '^' OPERATOR AND OP2 ARE OPTIONAL AS THEY ARE SHOWN WITHIN '()' BRACKETS. THE '^' IN THE EXAMPLE DENOTES A MATH-LIKE OPERATOR. THESE ARE:

ARITHMETIC:

+	(PLUS)
-	(MINUS)
*	(MULTIPLY; LO-BYTE RESULT)
**	(MULTIPLY; HI-BYTE RESULT)
/	(DIVIDE; INTEGER RESULT)
MOD	(DIVIDE; REMAINDER RESULT)

LOGICAL:

AND	(LOGICAL 'AND')
ORA	(LOGICAL 'OR')
EOR	(EXCLUSIVE 'OR')

EXTENDED LOGICAL:

+OR-	(RANDOMLY WILL ADD OR SUBT)

```
;       (2-VARIABLE SEPARATOR)
,       (OP1 INDEXED BY OP2)
%       (PLUS TIL OVERFLOW.
           MINUS TIL UNDERFLOW)
```

THE M-EXPRSN IS USED TO 'BIND' THE COUNT USED BY THE 'FOR' STATEMENT AND THIS IS ITS ONLY USE WITH CONTROL STATEMENTS. IT HAS MUCH GREATER USE WITH SETTING VARIABLES AS DESCRIBED IN THE NEXT CHAPTER.

CONCLUSION:

IF YOU HAVE GOTTEN THIS FAR AND FEEL THAT YOU UNDERSTAND IT ALL. CONGRATULATIONS. IF THINGS ARE A BIT HAZY OR EVEN IF YOU FEEL A LITTLE LOST, DONT SWEAT IT; ACTUAL EDITING WILL MAKE THINGS CLEAR.

A CEEMAC 'SCORE' IS COMPOSED OF A MAXIMUM OF 256 STATEMENTS. THE FIRST IS THE TITLE LINE AND THE LAST IS THE 'FENCE' STATEMENT. THESE AND THE CONTROL STATEMENTS (DO, GOTO, GOSUB, ETC) WERE DISCUSSED IN THE PRIOR CHAPTER. EDITING OF VARIABLES AND MACROS WILL BE EXAMINED HERE IN SOME DETAIL.

TO CORRECT ANY PART OF A STATEMENT, SIMPLY SCROLL RIGHT USING THE '-->' KEY. THE ARROW CURSOR FREEZES AND THE BLINKING BLOCK CURSOR TAKES OVER.

NOW STATEMENTS CAN BE EDITED BY OVER-TYPING AS WITH MOST EDITORS. WHEN 'ON THE LINE' THERE ARE THREE WAYS TO EXIT:

1- A 'C/R' CAUSES INPUT OF THE CHARACTER STRING TO THE LEFT OF THE CURSOR POINT. THIS IS STANDARD FOR MOST LINE EDITORS USED FOR COMPUTER LANGUAGES.

2- A CTRL-A CAUSES INPUT OF THE ENTIRE CHARACTER STRING ON THE LINE REGARDLESS OF THE THE POSITION OF THE CURSOR. THIS IS NOT GENERALLY A FEATURE OF LINE EDITORS AND IS VERY EFFICIENT IN SOME CASES.

3- A '<--' OR 'RETURN' AT THE VERY FIRST CHARACTER IS CONSIDERED A 'BACKOUT' AND LEAVES THE LINE AS IT WAS BEFORE STARTING TO EDIT IT. THIS LETS YOU PRESERVE THE ORIGINAL STATEMENT WHEN YOU'VE CHANGED YOUR MIND OR GOTTEN LOST. MOST LINE EDITORS PROVIDE AN EQUIVALENT 'BACKOUT' FEATURE.

INSERTING A NEW STATEMENT IS STARTED BY THE 'I' COMMAND WHEN AT THE BLINKING ARROW CURSOR. A BLANK LINE WILL OPEN UP JUST ABOVE THE CURRENT STATEMENT. SIMPLY TYPE THE NEW STATEMENT AND HIT THE RETURN KEY. CEEMAC ASSUMES YOU (MAY) WANT TO INSERT A SERIES OF NEW STATEMENTS AND AUTOMATICALLY PROVIDES A NEW BLANK LINE AFTER EACH INSERT. TO CEASE INSERTING, EXIT THE MODE BY A '<--' OR 'RETURN' WHILE AT THE FIRST CHARACTER POSITION.

AN ALTERNATE WAY TO CHANGE AN EXISTING STATEMENT IS TO HIT 'R' WHEN AT THE BLINKING ARROW CURSOR. THE ACTION IS LIKE THE '-->' KEY, EXCEPT THE OLD STATEMENT DISAPPEARS AND A BLANK LINE IS PROVIDED AS WITH THE 'I' INSERT COMMAND.

IT'S A GOOD IDEA TO TRY THESE MOVES ON A SAMPLE SCORE TO GET THE FEEL.

ALL VARIABLES IN CEEMAC ARE PRE-DEFINED. THAT'S BOTH THE BAD NEWS AND THE GOOD NEWS. COMPARED WITH GENERAL PURPOSE LANGUAGES, ITS BAD NOT TO BE ABLE TO NAME AND USE VARIABLES ANY WAY YOU WISH. WHATEVER THE ADVANTAGES, THIS HAS BEEN TRADED OFF IN CEEMAC TO PROVIDE SOME VERY POWERFUL EXECUTION AND COMPOSITIONAL FEATURES WHICH WILL BE EXAMINED IN DETAIL IN THE CHAPTER ON VARIABLES.

VARIABLES CAN BE SET IN CEEMAC BY A STATEMENT OF THIS FORMAT:

```
-->VAR1 = (M-EXPRSN)
```

THE STRUCTURE OF AN M-EXPRESN WAS DISCUSSED BRIEFLY IN THE PREVIOUS CHAPTER TO SHOW HOW IT WAS USED TO BIND THE LOOP COUNT FOR THE 'FOR' STATEMENT. IT IS THE SAME HERE, INCLUDING THE OPTIONAL SECOND OPERAND. THUS EITHER OF THESE EXAMPLES ARE VALID:

```
-->X1 = $67
-->X1 = V1 + $67
```

IMPORTANT! NOTE THE BLANK BEFORE AND AFTER THE '+' OPERATOR!

WHEN A VARIABLE IS TO THE LEFT OF THE '=', IT MAY BE REFERRED TO AS THE 'TARGET' VARIABLE. ALSO, NOTICE THAT THE FIRST EXAMPLE IS QUITE SIMPLE AND THE LITERAL ($67) COULD JUST AS WELL HAVE BEEN A VARIABLE.

THERE ARE SOME SPECIAL CASES WHERE TWO CONCEPTUALLY 'ADJACENT' VARIABLES CAN BOTH BE SIMPLY SET EQUAL TO LITERALS OR OTHER VARIABLES, LIKE SO:

```
-->XY1 = V1;$67
```

THIS '2-VARIABLE' STATEMENT IS PRIMARILY A SPACE SAVER AND THE 'XY1' IS CALLED A 'PSUEDO' VARIABLE. ONLY A SELECT FEW SUCH COMBINATIONS ARE PERMITTED

AND THEY ARE DETAILED ELSEWHERE. NOTE THE ';' USED TO SEPARATE OP1 FROM OP2. IT IS CALLED A 'DELIMITER' AND SERVES THE SAME PURPOSE AS A BLANK; IN FACT, IT IS OKAY TO ENTER THE ABOVE EXAMPLE IN COMPACT FORM LIKE SO:

```
-->XY1 V1 67
```

THE EDITOR WILL UNDERSTAND AND REDISPLAY IN THE PRIOR EXAMPLE FORMAT.

IF VARIABLES ARE THE 'JEWELS' OF CEEMAC, THEN MACROS ARE THE WORKHORSES. THEY LOOK LIKE THIS:

```
-->BLINE [V3;$F]
-->SWAP [V1;V3]
```

IN THE ABOVE EXAMPLES, THE '[', ']', AND '=' CHARACTERS ARE OPTIONAL; THEY CAN BE ENTERED OR OMITTED (OR CURSED OVER). CEEMAC WILL ALWAYS REGENERATE THE STATEMENT IN STANDARD FORMAT. AGAIN, A BLANK DELIMITER CAN BE USED IN LIEU OF THE ';' CHARACTER. SIMPLIFY YOUR ENTRY, IF YOU WISH, AS FOLLOWS:

```
-->BLINE V3 F
```

MACROS CAN TAKE ONE OR TWO PARAMETERS (CALLED 'PARAM1' AND 'PARAM2'). IF EITHER OR BOTH ARE OMITTED, THE EDITOR WILL ASSUME THEM TO BE '0'. BEFORE USING A MACRO, THE DOCUMENTATION DEALING WITH IT SHOULD BE READ WITH PARTICULAR ATTENTION PAID TO THE USE OF PARAMETERS AND PRESETS.

A DELIMITER IS ALWAYS REQUIRED BETWEEN TWO FIELDS. A PARTICULARLY HARD POINT TO REMEMBER IS THAT THE MATH-A-LOGOCAL OPERATORS ARE CONSIDERED 'FIELDS' IN CEEMAC. THIS PROTOCOL CAN FEEL UNNATURAL AND MIGHT BE CHANGED IN THE NEXT RELEASE. MEANWHILE, NOTE THAT THE FOLLOWING WILL BE REJECTED BY THE EDITOR:

```
--->X1 = V1+$67
```

DO IT AS SHOWN PREVIOUSLY.

LITERALS CAN BE ENTERED IN ANY OF THE FOLLOWING WAYS:

#34 (= $22)

$22 (= #34)

22 (= $22 IF IN HEX MODE)
(= #22 IF IN DECIMAL MODE)

THE EDITOR REMEMBERS THE MODE OF YOUR LAST ENTERED LITERAL AND USES THAT MODE TO DISPLAY ALL LITERALS. IF THE MODE CHANGES, THE PORTION OF THE SCORE ON DISPLAY WILL BE REEDITED IN THE NEW MODE IMMEDIATELY.

SYNTAX ERRORS ARE DETECTED WHEN A STATEMENT EDIT IS FINISHED. THE CURSOR IS POSITIONED OVER THE FIRST CHARACTER OF THE OFFENDING WORD OR FIELD.

CONSISTANCY AND COMPLETENESS ERRORS CANNOT BE DETECTED UNTIL THE SCORE EDIT IS COMPLETE AND EXECUTION ATTEMPTED. FOR EXAMPLE, THE VALIDITY OF A 'GOTO 3' STATEMENT CANNOT BE KNOWN AT ENTRY TIME SINCE THE SYMBOL '3' MAY BE ENTERED OR REMOVED AFTER THE 'GOTO' STATEMENT.

HOWEVER, SUCH ERRORS ARE DETECTED BEFORE SCORE EXECUTION IS ALLOWED. THE APPROPRIATE MESSAGE IS DISPLAYED IN THE BOTTOM WINDOW WITH THE CURSOR POINTING TO THE LINE AT THE POINT OF DETECTION (WHICH MAY OR MAY NOT BE THE BAD STATEMENT). A LIST OF THESE MESSAGES FOLLOW:

DUPLICATE SYMBOL

REFERENCE IS UNDEFINED

GIVEN SHAPE# IS NOT IN SHAPE TABLE

PARAM1 MUST BE A VARIABLE

PARAM1 AND PARAM2 MUST BOTH BE VARIABLES

NO 'DO/FOR' STMT FOR THIS 'AGAIN' STMT

MISSING 'AGAIN' STMT (OR EXTRA 'DO/FOR')

'SUB' MAY NOT START WITHIN A LOOP

'GOTO'S MAY NOT EXIT A LOOP OR SUB

SYMBOLS 0, 1 & F NOT PERMITTED IN LOOPS

LOOP/SUB NEST IS TOO DEEP (MAX=15)

EXCESS SUBS (MAX=64)

SCORE AREA FULL; MAY EXECUTE AS LISTED

REPOOL PARAMS INVALIDITY; (P1)= PF)

OBVIOUS STRUCTURE ERROR

SYSTEM ERROR

CTRL-A WILL CAUSE THE SCORE TO EXECUTE (IF FOUND TO BE VALID). CTRL-C WILL EXIT TO DOS AND CAUSE 'BSAVE' LINES TO BE CREATED TO AID IN SAVING THE SCORE, THE SHAPE TABLE OR THE 'LISTS' MODULE TO DISK.

--- TO PRINT YOUR SCORE ---

PRINTER IN-	KEY TO PRINT-
SLOT-1	!
SLOT-2	"
SLOT-3	#
SLOT-4	$
SLOT-5	%
SLOT-6	&
SLOT-7	'

SCREEN CONCEPTS

THE ABILITY TO 'VISUALIZE' IN GRAPHIC TERMS IS QUITE IMPORTANT TO SUCCESS IN CREATING PLEASING ABSTRACTIONS NO MATTER WHAT LANGUAGE YOU USE. EVERY EFFORT HAS BEEN MADE TO ASSIST THE VISUAL COMPOSER IN THIS BY DESIGNING CEEMAC WITH THE FOLLOWING ATTRIBUTES:

- 'TRIAL AND WOW' DEVELOPEMENT
- INTELLIGENT DEFAULTS
- NATURAL KEYING
- SIMPLIFIED STRUCTURES
- MULTIPLE GROWTH PATHS

THE APPLE][COMPUTER HAS A HIGH RESOLUTION SCREEN THAT CONTAINS 280 HORIZONTAL POINTS BY 192 VERTICAL POINTS. HOWEVER, CEEMAC HAS ADOPTED A COORDINATE SYSTEM BASED ON 8-BIT VALUES WHERE THE 'UNIT SCREEN' IS 256 X 256 POINTS. HOW THIS MAPS TO THE APPLE SCREEN WILL BE SHOWN LATER.

INTERNALLY, CEEMAC KEEPS ITS OWN 24-BIT 'UNIVERSAL' SCREEN AND MAPS EVERYTHING TO IT. THIS PROVIDES A METHOD TO 'CLIP' LINES AND SHAPES THAT WANDER OFF OUR UNIT SCREEN. IT ALSO PREVENTS ROUNDING ERRORS FROM ACCUMULATING AND KEEPS THE DOOR OPEN FOR THE HIGHER RESOLUTIONS OF THE FUTURE THAT ARE CERTAIN TO COME.

TRADITIONALLY, COORDINATE SYSTEMS HAVE EITHER SET 0,0 AT CENTER OR LOWER LEFT. THE APPLE][SETS 0,0 AT UPPER LEFT (WHY, IS A SEPARATE STORY). OUR 256 X 256 UNIT SCREEN SETS 0,0 AT LOWER LEFT AND IN THIS RESPECT DIFFERS FROM THE APPLE STANDARD. (IT DOES CONFORM, HOWEVER, WITH AT&T'S VIDEOTEX STANDARD).

THE 'ASPECT RATIO' OF A SCREEN IS ITS WIDTH COMPARED TO ITS HEIGHT. FOR OUR PURPOSES, WE WILL USE THIS TERM TO APPLY TO THE RATIO OF X AND Y COORDINATES RATHER THAN ANY VIEWABLE RATIO THAT COULD BE MEASURED ON THE FACE OF THE TV/CRT SCREEN ACUALLY BEING USED.

AS PREVIOUSLY STATED, THE APPLE SUPPORTS A HORIZONTAL RESOLUTION OF 280 POINTS. EXCEPT FOR 'FULL SCREEN' COMMANDS (LIKE 'CLEAR'), THE LEFTMOST 12 POINTS AND THE RIGHTMOST 12 POINTS ARE IGNORED BY CEEMAC AS IF THEY DONT EXIST! THIS REDRESSES THE VIEWABLE HORIZONTAL GRID TO A MAXIMUM OF 256 POINTS.

APPLE'S VERTICAL RESOLUTION IS 192 POINTS (IGNORING MIXED GRAPHICS MODE) WHICH GIVES US A 'TARGET' SCREEN OF 256 X 192 (A 4:3 ASPECT RATIO). AS A DEFAULT STANDARD, CEEMAC WILL SIMPLY MAP THE 256 X 256 UNIT SCREEN TO THE 256 X 192 APPLE SCREEN AS JUST DEFINED. HOWEVER, CEEMAC WILL NOT NORMALLY PLOT ANY POINT WHICH HAS EITHER AN X OR Y COORDINATE OF ZERO. (THE REASONS FOR THIS SEEMINGLY ARBITRARY TRUNCATION ARE TOO COMPLEX TO GO INTO HERE).

WHILE A 4:3 ASPECT RATIO IS FAIRLY NORMAL, OTHERS SUCH AS 3:2, 5:4, 5:3 AND, EVEN, 2:1 ARE USED IN DIFFERENT GRAPHICS HARDWARE. THEORETICALLY, WHATEVER THE 'NATIVE' ASPECT RATIO OF A PARTICULAR GRAPHICS DISPLAY, CEEMAC WILL REGARD IT AS THE WORKING STANDARD AND MAP ITS UNIT SCREEN ACCORDINGLY.

THE POINT OF EXPLAINING ALL THIS IS THAT WE HAVE THE ABILITY TO DISTORT THIS 'NORMAL' RATIO USING A CEEMAC MACRO CALLED 'SETASP'. ROUGHLY, IT WORKS THIS WAY:

CODE	ACTION ON THE UNIT SCREEN
$00	SQUEEZE Y-COORD TO SCREEN
$01	CLIP Y-COORD EQUALLY TOP & BTM (NO SQUEEZING)
$02	SQUEEZE X & Y-COORDS EQUALLY
$03	REVERSE NORMAL RATIO MAPPING

CODE $00 SIMPLY MAPS THE STANDARD RATIO FOR THE TARGET SCREEN.

CODE $01 AVOIDS THE DISTORTION INTRODUCED IN THE Y-SQUEEZE BY CLIPPING POINTS AT THE SCREEN TOP AND BOTTOM.

CODE $02 SQUEEZES EQUALLY THE X- AND Y-COORDINATES THEREBY AVOIDING DISTORATION AT THE COST OF REDUCED OVERALL SIZE.

CODE $03 REVERSES THE 'NORMAL' DISTORTION INTRODUCED BY THE RATIO OF THE TARGET SCREEN BY EXTRA SQUEEZING OF THE X-COORDINATE.

OTHER CODES ARE RESERVED FOR FUTURE UNDEFINED ASPECT SOLUTIONS.

THE RATIONALE FOR THIS SYSTEM LIES IN THE ABILITY OF THE COMPOSER TO WORK WITH THE UNIT SCREEN WHILE IGNORING THE ACTUAL TARGET SCREEN COORDINATES OR ASPECT RATIO. SCORE COORDINATES CAN BE MAPPED TO NON-APPLE TARGETS WITH MINIMUM DISTORTION OF THE INTENDED PICTURE.

COLORS

THEORY (AND FANTASY):

COLORS DONT NEED NAMES. ASSIGN THEM A VALUE FROM 0 TO $FF AND LET FLY. THEN 'ROTATE' YOUR SELECTED COLOR AROUND A 256-PART CIRCLE TO IMITATE YOUR TV'S 'TINT' KNOB AND YOU'LL BE SELECTING COLOR IN A MORE MEANINGFUL WAY THAN YOU EVER COULD HAVE BY SAYING OR WRITING A WORD (BLUE BY ANOTHER NAME IS STILL BLUE).

PRACTICE:

THERE ARE SOME PROBLEMS ACTUALIZING THE ABOVE ON WHAT IS ESSENTIALLY A 'PSUEDO COLOR' APPLE. IN FACT, IT CAN'T BE DONE. THERE'S ONLY THE 4 COLORS (SINCE WHEN ARE BLACK AND WHITE REALLY COLORS?) AND THEIR AVAILABILITY IS CONSTANTLY BEING TRADED OFF AGAINST POSITION, RESOLUTION AND COMPLEXITY.

THE DIGITAL 'TINT KNOB' IS, UNFORTUNATELY, UNAVAILABLE ANYWHERE (THAT I KNOW OF) AND CERTAINLY NOT ON THE APPLE. ALSO, WITH THE APPLE, HALF THE 'COLORS' DONT SHOW ON HALF OF THE X-COORDINATES. AND CONFLICTS CAUSED BY THE 'PHASE BIT' CAN CREATE QUITE DISAGREEABLE 'BLOCKING' EFFECTS.

SO HERE IS THE PRESENT (NON) SOLUTION:

SPECIFY ONE OF 4 COLORS AS FOLLOWS:

```
$20= VIOLET (APPLE MASK= $55)
$60= BLUE   (APPLE MASK= $D5)
$A0= GREEN  (APPLE MASK= $2A)
$E0= ORANGE (APPLE MASK= $AA)
```

AND:

```
$00= BLACK  (APPLE MASK= $00)
$FF= WHITE  (APPLE MASK= $7F)
```

EXAMPLE: 'COLOR = $60' GETS BLUE

THE CODES SELECTED FOR THE COLORS ALLOW BOOLEAN MANIPULATIONS WHICH CAN RESULT IN HARMONIOUS HUE RELATIONSHIPS (THOUGH NOT NECESSARILY ON THE APPLE). NOTE THAT THE IMPLEMENTATION CRUDENESS MASKS A FUNDAMENTAL PRINCIPLE. NAMELY, CONVERTING THE VISIBLE SPECTRUM FREQUENCIES AROUND THE COLOR CIRCLE TO DIRECT DIGITAL VALUES SEEMS TO BE THE LOGICAL APPROACH FOR THIS NEW TECHNOLOGY. (WILL BLUE STILL BE BLUE?)

SO WHAT CAN BE DONE ABOUT THIS 'PSUEDO COLOR' MACHINE? SIMPLY GO AHEAD ON INSTINCT. IT'LL PROBABLY LOOK PRETTY GOOD EVEN IF YOU DONT HAVE A MINDBOGGLING THEORY BEHIND IT.

VARIABLES

VARIABLES IN CEEMAC ARE PRENAMED AND PREDEFINED FOR ONE OR MORE PURPOSES. THEIR NAMES NEVER EXCEED 6 CHARS. THEIR VALUES ARE LIMITED TO 8 BITS (0-255). THEY APPEAR AS PARAMETERS IN MACRO STATEMENTS, OP1 OR OP2 VALUES IN C-EXPRSNS AND M-EXPRSNS. THEY MAY ALSO BE THE 'TARGET' OF A T-EXPRSN AS IN THIS EXAMPLE:

```
-->TVAR = OP1 ( ^ OP2 )
```

USUALLY, A LITERAL MAY BE USED ANYWHERE THAT A VARIABLE IS PERMITTED EXCEPT AS THE TARGET IN A T-EXPRSN.

AT THE START OF EXECUTION, ALL VARIABLES (AND SWITCHES) ARE INITIALIZED ZERO EXCEPT THE FOLLOWING:

```
COLOR = $FF (WHITE)
RANDOM VARIABLES (RANDOM)
'POOL' VARIABLES (RANDOM)
X1,Y1 = $80,$80 (SCREEN CENTER)
X2,Y2 = RANDOM (RANGE= 0-$FF)
XSCALE = 1; YSCALE = 1
REVCNT = 1
```

A DETAILED LOOK AT ALL CURRENTLY AVAILABLE VARIABLES FOLLOWS:

COLOR:

```
$00= BLACK   (APPLE MASK= $00)
$20= VIOLET  (APPLE MASK= $55)
$60= BLUE    (APPLE MASK= $D5)
$A0= GREEN   (APPLE MASK= $2A)
$E0= ORANGE  (APPLE MASK= $AA)
$FF= WHITE   (APPLE MASK= $7F)
```

(ALSO, SEE 'COLOR' CHAPTER)

COORDINATE SETS

```
X0/Y0: GHOSTPOINT FOR SPLINE
       POSSIBLE POINT FOR ADOT
       UPPER RIGHT POINT FOR ABOX
         (IF A QUADRANGLE)

X1/Y1: START POINT FOR BLINE
       START POINT FOR SPLINE
       START POINT FOR SHAPE
       POSSIBLE POINT FOR ADOT
       UPPER LEFT POINT FOR ABOX

X2/Y2: END POINT FOR BLINE
       END POINT FOR SPLINE
       POSSIBLE POINT FOR ADOT
       LOWER RIGHT POINT FOR ABOX

X3/Y3: GHOSTPOINT FOR SPLINE
       POSSIBLE POINT FOR ADOT
       LOWER LEFT POINT FOR ABOX
         (IF A QUADRANGLE)

  XY0: 2-BYTE 'PSUEDO' VARIABLE
  XY1:   "         "        "
  XY2:   "         "        "
  XY3:   "         "        "
```

FREE-USE VARIABLES:

V1, V2...V9, VA...VF

HAVING NO DEDICATED OR IMPLIED USE, THESE 15 VARIABLES ARE AVAILABLE TO THE COMPOSER TO USE ANY WAY HE CHOOSES.

RANDOM VARIABLES (READ ONLY):

ARE RESET TO AN UNPREDICTABLE VALUE WITHIN THE RANGE SHOWN EVERY TIME USED (EXCEPT IN TRACE STATEMENTS).

RND1: RANGE= 0-1

RND2: RANGE= 0-3

RND3: RANGE= 0-7

RND4: RANGE= 0-$F

RND5: RANGE= 0-$1F

RND6: RANGE= 0-$3F

RND7: RANGE= 0-$7F

RANDOM: RANGE= 0-$FF

'POOL' VARIABLES (NORMALLY READ ONLY):

THESE VARIABLES CAN BE SET (AND RESET) TO RANDOM VALUES BY THE 'NUPOOL' MACRO. THEY ARE NOT RESET EACH TIME REFERENCED AS ARE THE RANDOM VARIABLES. RESTRICTING THE LOWER LIMIT TO 1 HAS PARTICULAR VALUE WHEN USING POOL VARIABLES AS INCREMENTERS OR SYMMETRY PARAMETERS.

P1: RANGE= 0-1

P2: RANGE= 1-3

P3: RANGE= 1-7

P4: RANGE= 1-$F

P5: RANGE= 1-$1F

P6: RANGE= 1-$3F

P7: RANGE= 1-$7F

P8: RANGE= 1-$FF

P9: RANGE= 1-$FF

:
:

PF: RANGE= 1-$FF

NOTE: THEY CAN ALSO BE USED AS FREE VARIABLES IN WHICH CASE THE RANGE LIMITS INDICATED ABOVE CANNOT BE ENFORCED. IN THIS CASE, IF THE 'NUPOOL' MACRO IS ALSO TO BE USED, CARE SHOULD BE TAKEN WHEN SETTING THE PARAMS TO EXCLUDE THOSE POOL VARIABLES BEING USED AS FREE VARIABLES.

'FINGER' VARIABLES (READ ONLY):

THE VALUES ARE SET BY THE 'PERFORMER' BY HITTING BUTTONS OR KEYS OR CHANGING PADDLE SETTINGS. SINCE THESE ARE EXTERNALLY CONTROLLED, COMPOSERS SHOULD AVOID TRYING TO 'SET' THEM WITHIN A SCORE.

PDL0: RANGE= 0-255

PDL1: " "

PDL2: " "

PDL3: " "

BUT0: PLUS OR MINUS

BUT1: " "

BUT2: " "

KEY: LATEST INVALID KEYIN
(ASCII; HI BIT ON)

'EXTRACTED' VARIABLES (READ ONLY):

THESE ARE EXTRACTED FROM THE CURRENT 'PDL1' READING. IN EFFECT, THEY ARE THE 'POWERS-OF-2' LOGICAL 'AND'ING OF PDL1 AT THE TIME OF REFERENCE.

EXT1: RANGE= 0-1 (HI OR LOW HALF)

EXT2: RANGE= 0-3 ('QUARTERS' PDL1)

EXT3: RANGE= 0-7 (CUT INTO EIGHTHS)

EXT4: RANGE= 0-$F (INTO SIXTEENTHS)

EXT5: RANGE= 0-$1F (ETC.)

EXT6: RANGE= 0-$3F (ETC.)

EXT7: RANGE= 0-$7F

EXT8: RANGE= 0-$FF

TABLE DRIVEN VARIABLES (READ ONLY):

THESE VARIABLES ARE SUPPORTED BY INTERNAL LOOKUP TABLES. THEY MIGHT BE CONSIDERED CEEMAC 'FUNCTIONS'.

NXTCOL: PICKS THE NEXT NON-BLACK/WHITE COLOR FROM THE HIRES COLOR TABLE.

RNDCOL: PICKS A NON-BLACK/WHITE COLOR FROM THE HIRES COLOR TABLE AT RANDOM.

NOTE: THERE ARE 4 COLORS IN THE APPLE COLOR TABLE IN THE ORDER: VIOLET, BLUE, GREEN AND ORANGE.

NXTPRM: PICKS THE NEXT PRIME NUMBER FROM THE PRIMES TABLE.

RNDPRM: PICKS A PRIME NUMBER FROM THE PRIMES TABLE AT RANDOM.

NOTE: THE PRIMES TABLE CONTAINS ALL PRIME NUMBERS FROM 3 TO 255. THERE ARE 51 IN ALL.

USER OPTIONS:

XOPT: EACH BIT HAS ITS OWN MEANING. UNDEFINED BITS MUST BE LEFT 'OFF' TO INSURE INTERNAL INTEGRITY. TO TURN ON' MORE THAN ONE BIT, THE COMPOSER MUST 'OR' BITS. FOR EXAMPLE, TO TURN ON BIT02, BIT10 & BIT20:

-->XOPT = $32

BIT02= DONT DRAW SHAPES OR BOXES FLATOUT (OBEY SPEED SETTINGS BETWEEN LINE SEGMENTS).

BIT08= DEFEAT THE N/256THS REMAINDER OF THE 'MOD' FUNCTION WHEN DIVIDING.

BIT10= NO SOUND FROM THE APPLE SPEAKER (USEFUL WHEN PLAYING 'ANOTE'S THRU EXTERNAL AUDIO SYSTEM).

BIT20= NO 'CLICK' ON INVALID KEYIN (USEFUL WHEN USING SUCH KEYS AS EXECUTION COMMANDS OR TO MUTE MUSIC).

LINETX: MEANS 'LINE TEXTURE'. THIS VARIABLE CARRIES PRESENT AND FUTURE OPTION BITS TO ALTER THE CHARACTERISTICS OF A STRAIGHT LINE. ONLY ONE OPTION IS DEFINED IN RELEASE 1.1 OF CEEMAC. DOTTED, DASHED, AND THICKER LINES ARE ANTICIPATED FUTURE OPTIONS.

BIT04= DO NOT OVERLAP 'STAIRSTEPPED' LINES. DIAGONAL LINES IN APPLE HIRES ARE NORMALLY DRAWN WITH OVERLAP AT THE 'STAIRSTEPS'. USUALLY THIS IS DESIREABLE BUT MAY NOT BE AS WHEN DRAWING SHAPE-DRIVEN TEXT CHARACTERS.

MUSIC VARIABLES (READ ONLY):

MUSESW: 0 OR NON-ZERO (OFF OR ON)

SOUND: LATEST CONTINUOUS DETECTION COUNT

QUIET: LATEST CONTINUOUS DETECTION COUNT

SOUND1: PRIOR 'SOUND' COUNT

QUIET1: PRIOR 'QUIET' COUNT

SOUND2: PRIOR 'SOUND1' COUNT

QUIET2: PRIOR 'QUIET1' COUNT

:

SOUND7: PRIOR 'SOUND6' COUNT

QUIET7: PRIOR 'QUIET6' COUNT

(SEE 'MUSIC' CHAPTER)

MATH VARIABLES (READ ONLY):

OTHER8: AFTER A MULIPLICATION OR DIVISION OPERATION, THIS VARIABLE ALWAYS CONTAINS THE 'OTHER' 8 BITS OF THE 16-BIT ANSWER.

MATHSW: AFTER ANY MATHEMATICAL OPERATION, THIS SWITCH IS SET OFF IF THERE IS NO OVERFLOW OR CARRY. IT IS SET ON IF THERE WAS OVERFLOW OR CARRY.

(SEE 'MATH-A-LOGICAL' CHAPTER)

'REVERSING' VARAIBLES:

REVCNT: USER SETABLE (DEFAULT= 1)

REVRSW: RESET TO REVCNT WHENEVER BUT0 IS PRESSED AND NO SYM-0 IN SCORE.

(SEE 'REVERSING' CHAPTER)

LISTS 'VARIABLES' (READ ONLY):

THESE ARE NOT, STRICTLY SPEAKING, VARIABLES. A DIRECT REFERENCE WILL SIMPLY GET YOU THE FIRST VALUE IN THE LIST. 'INDEXED' REFERENCE IS THE INTENDED USE.

LIST1: A 256-BYTE LIST OF SINE WAVE VALUES THAT CAN SERVE AS COORDINATES, INCREMENTERS, SCALERS, ETC. EXAMPLE:

--)Y1 = LIST1 , V1

LIST2: THIS 256-BYTE LIST CONTAINS A 'DAMPING' SINE WAVE IF PLOTTED ACROSS THE SCREEN LIKE THIS:

```
   X1 = 1
   DO
----)Y1 = LIST2 , X1
     ADDT [1;0]
     X1 = X1 + 1
   AGAIN UNLESS X1 = 0
```

LIST3: HAS 128 POINTS ON A CIRCLE (X,Y) AND CAN BE USED TO DRAW A DOTTED CIRCLE LIKE SO:

```
   XY1 = 0;0
   V1 = 0
   FOR $80
     X1 = LIST3 , V1
     V1 + 1
     Y1 = LIST3 , V1
     ADDT [1;0]
     V1 = V1 + 1
   AGAIN
```

LIST4: HAS 128 POINTS ON A SPIRAL (X,Y). DRAW A SPIRAL LIKE USING THE CIRCLE EXAMPLE ABOVE, EXCEPT SUBSTITUTE 'LIST4' FOR 'LIST3'.

(SEE 'LISTS' CHAPTER)

SHAPES SUPPORT:

ROTATE: 256 POINTS ON THE CEEMAC 256-POINT COMPASS

```
       -( UPSCALERS )-
XSCALE: HORIZONTAL MULTIPLIER
        ($00 DEFAULTS TO $01)
YSCALE: VERTICAL MULTIPLIER
        ($00 DEFAULTS TO $01)
XYSCAL: 2-BYTE 'PSUEDO' VARIABLE
```

NOTE1: ISCALE & DSCALE ARE CONVENIENCE MACROS (NOT VARIABLES)

NOTE2: SSIZE IS A DOWNSCALER MACRO FOR USE WITH SHAPES (SEE THE 'SHAPES' CHAPTER)

SPLINE VARIABLES:

DENSTY: AN OPTIONAL VALUE TO INCREASE OR DECREASE THE NUMBER OF POINTS ON A SPLINE. IF OMITTED, CEEMAC USES AN INTERNAL CALCULATION TECHNIQUE TO DETERMINE LINE DENSITY.

(SEE 'SPLINES' CHAPTER)

MATH-A-LOGICAL

CEEMAC IS NOT HEAVILY MATH ORIENTED (AND NEITHER AM I). MOST OF THE POSSIBLE OPERATIONS ARE SIMPLE AND WELL KNOWN. SOME OF THE VARIATIONS AND SPECIAL TWISTS NEED A LITTLE EXPLAINING. IN WHAT FOLLOWS, OP1 IS THE OPERAND BEFORE THE MATHLIKE OPERATOR AND OP2 IS THE OPERAND AFTER. OP1 AND/OR OP2 CAN BE EITHER VARIABLES OR LITERALS. THIS IS CALLED AN 'M-EXPRSN'. IN THE TRIVAL CASE, AN 'M-EXPRSN' WOULD CONTAIN ONLY OP1 AND, THEREFORE NO OPERATOR NOR OP2 (EG. V1 = X2).

THERE ARE TWO VARIABLES CALLED 'MATHSW' AND 'OTHER8' WHICH CONTAIN AUXILLIARY INFORMATION WHENEVER THESE FUNCTIONS ARE PERFORMED.

+ (PLUS) QUITE SIMPLY, ADDS OP1 TO OP2 PUTTING THE RESULT IN THE TARGET VARIABLE. THE MATHSW IS SET 'ON' IF OVERFLOW OCCURRED.

- (MINUS) SUBTRACTS OP2 FROM OP1, AGAIN REPLACING THE TARGET VARIABLE WITH THE RESULT. MATHSW IS SET 'ON' IF UNDERFLOW.

+OR- RANDOMLY EITHER ADDS (LIKE +) OR SUBTRACTS (LIKE -). USEFUL WHEN A RANDOM ACTION DESIRED. MATHSW IS SET 'ON' IF EITHER OVERFLOW OR UNDEFLOW. THIS GUY IS DESIGNED PRIMARILY FOR SMALL INCREMENTS (1 IS TYPICAL). IF A ZERO INCREMENT IS SPECIFIED, IT IS FORCED TO 1 (DURING EXECUTION). ALSO, WRAPAROUND IS AUTOMATICALLY PROHIBITED.

% (PERCENT SIGN) ADD TIL OVERFLOW, SUBTRACT TIL UNDERFLOW. THE PRIMARY USE IS TO PREVENT ANNOYING WRAPAROUND WHEN INCREMENTING A SCREEN COORDINATE. CAUSES A 'BOUNCE' OR REVERSAL OF THE PATH WHEN IT REACHES THE SCREEN EDGE (SEE CHAPTER ON REVERSING). MATHSW IS SET 'ON' IF OVERFLOW OR UNDERFLOW.

* MULTIPLY OP1 TIMES OP2 DELIVERING THE LO-BYTE (8 BITS) RESULT TO THE TARGET VARIABLE. THE HI-BYTE RESULT CAN BE FOUND IN THE 'OTHER8' VARIABLE.

** MULTIPLY OP1 TIMES OP2 PUTTING THE HI-BYTE RESULT IN THE TARGET VARIABLE. 'OTHER8' GETS THE LO-BYTE.

/ DIVIDE OP1 BY OP2 SENDING THE INTEGER RESULT TO THE TARGET VARIABLE WHILE THE REMAINDER GOES TO 'OTHER8'.

MOD DIVIDE OP1 BY OP2 WITH THE REMAINDER GOING TO THE TARGET VARIABLE AND 'OTHER8' GETTING THE INTEGER RESULT.

SPECIAL NOTE ON REMAINDERS: CEEMAC NORMALLY DELIVERS A REMAINDER IN TERMS OF 'N/256THS' OF 1. CONCEPTUALLY, THIS IS THE HEX EQUIVALENT OF DECIMAL REMAINDERS WHICH ARE IN N/100THS, N/1000THS, ETC. IT TAKES THE CEEMAC INTERPRETER A LITTLE LONGER TO DO THIS AND THERE MAY BE REASONS WHY YOU WILL PREFER THE REMAINDER IN N/DIVISOR (A FRACTION). IF SO SET THE XOPT/BIT0B ON IN THE SCORE TO GET ALL '/' AND 'MOD' OPERATIONS IN THE LATTER FORM. OR TURN IT ON OR OFF WHEREVER DESIRED.

AND THIS IS THE BOOLEAN LOGICAL 'AND'. BIT POSITIONS FROM OP1 WILL BE TURNED OFF IF THE COORESPONDING BITS IN OP2 ARE OFF. OTHER BITS WILL BE UNAFFECTED AND THE RESULT GOES TO THE TARGET VARIABLE (AS USUAL). HANDY FOR RESTRICTING A VALUE TO A LIMITED RANGE. MATHSW IS ON IF RESULT DIFFERENT THAN OP1.

ORA THE BOOLEAN LOGICAL 'OR' WILL HAVE BITS ON IN THE RESULT IF EITHER THE OP1 OR OP2 BIT WAS ON. GOOD FOR FORCING A VALUE ODD, ETC. MATHSW IS ON IF RESULT DIFFERENT THAN OP1.

EOR THIS IS THE LOGICAL 'EXCLUSIVE OR' WHICH REVERSES THE OP1 BIT WHEREVER THE CORRESPONDING BIT IN OP2 IS ON. OP1 IN NOT ACTUALLY EFFECTED; THE RESULT GOES TO THE TARGET VARIABLE AS IN ALL THESE CASES. MATHSW IS ON IF RESULT DIFFERENT THAN OP1.

; (SEMI-COLON) THIS SERVES ONLY THE SINGLE PURPOSE OF SEPARATING OP1 FROM OP2. IT APPEARS IN 2-BYTE PSUEDO VARIABLES AND MACROS.

, (COMMA) DEFINES OP1 AS THE START OF A LIST AND OP2 AS THE OFFSET FROM THAT START. TECHNICALLY, THIS IS CALLED 'INDEXING' AND IS USEFUL IN EXTRACTING A PARTICULAR OR NEXT BYTE FROM A LIST.

LIKE MOST EVERYTHING ELSE IN CEEMAC, THE EASIST WAY TO UNDERSTAND IS TO TRY IT OUT. WHERE YOU'RE UNSURE OF THE CONCEPT, SIMPLY CREATE A SMALL LOOP USING THE OPERATOR AND INCLUDE A TRACE MACRO TO DISPLAY THE VALUES AS THEY CHANGE.

REVERSING

THIS SEPARATE EXPLANATION ON REVERSING IS NECESSARY BECAUSE ITS A CONCEPT THAT SPREADS OVER SEVERAL AREAS OF CEEMAC. ITS ALMOST AN OCCULT FUNCTION AND THE NEW COMPOSER WOULD BE WISE TO SIMPLY IGNORE IT IF THE FIRST READING IS CONFUSING (AS IT PROBABLY WILL BE). YOU CAN ALWAYS RETURN TO THIS WHEN YOU'RE 'UP TO SPEED' AND IT WILL ALL MAKE BETTER SENSE THEN.

TO REVIEW A LITTLE FROM THE 'MATH-A-LOGICAL' CHAPTER, THE '%' OPERATOR CAN BE USED INSTEAD OF A '+' OR A '-' IN M-EXPRSNS TO AVOID WRAPAROUND. YOU WILL RECALL THAT THE '%' GUY WORKS EXACTLY LIKE A '+' OPERATOR UNTIL THERE IS OVERFLOW, AT WHICH POINT IT ACTS LIKE A '-' OPERATOR UNTIL THERE IS UNDERFLOW, AND SO ON.

THE RESULT OF THIS WRAPAROUND AVOIDANCE IS TO 'REVERSE THE PATH' THUS CAUSING A SORT OF REFLECTION EFFECT AT THE SCREEN EDGE. IT MIGHT BE NICE TO BE ABLE TO DO THIS AT WILL DURING EXECUTION. THE RETURN KEY (SEE 'EXECUTION COMMANDS') CAN DO JUST THAT PROVIDING THE SCORE HAS ANTICIPATED THE POSSIBILITY. THIS IS FACILITATED BY TWO QUITE SPECIAL VARIABLES WHICH ARE NOW DESCRIBED.

PLEASE PLAN ON RE-READING THE NEXT THREE PARAGRAPHS BECAUSE THE LOGIC IS NECESSARILY INTERWOVEN.

'REVCNT' MEANS 'REVERSE COUNT'. THE COMPOSER SETS THIS VALUE USUALLY NEAR THE START. THE DEFAULT SETTING IS '1'. WHENEVER THE INTERPRETER DETECTS THAT THE RETURN KEY HAS BEEN HIT, IT LOADS THE 'REVRSW' VARIABLE WITH THE COUNT FROM REVCNT.

WHEN THE INTERPRETER EXECUTES A STATEMENT WITH THE '%' OPERATOR, REVRSW IS FIRST TESTED FOR ZERO. IF NOT ZERO, REVRSW IS DECREMENT BY ONE AND THE PATH CHANGED IMMEDIATELY BY FLIPPING TO MINUS IF THE STATEMENT WAS 'PLUSING' OR TO PLUS IF THE STATEMENT HAD BEEN 'MINUSING'.

A LITTLE THOUGHT WILL REVEAL THAT IF THE REVCNT IS SET TO 2 AND THE RETURN KEY IS HIT, THE NEXT TWO STATEMENTS ENCOUNTERED BY THE INTERPRETER USING THE '%' OPERATOR WILL BE FORCED TO REVERSE THEIR +/- CONDITION. IF THERE IS ONLY A SINGLE STATEMENT USING THE % OPERATOR, THE PATH BEING DRAWN BY THIS STATEMENT WILL REVERSE ITSELF TWICE, DRAWING A 'HITCH' IN THE LINE (OR WHATEVER IT IS DOING).

IF AFTER REREADING THIS EXPLANATION YOU ARE UNCLEAR, YOU MIGHT EXPERIMENT WITH THE FOLLOWING TEST SCORE.

```
SCORE:   TEST REVERSING CONCEPT
  CLEAR [0;0]
  XY0 = RANDOM;RANDOM
  DO
    ADOT [0;0]
    Y0 = Y0 % 3
    X0 = X0 % 1
  AGAIN
```

THIS SCORE SHOULD DRAW A PATTERNED MOSAIC BY 'BOUNCING' OFF THE SCREEN EDGES. RESTART THE SCORE AND WATCH WHAT HAPPENS WHEN YOU HIT THE RETURN KEY. REVERSALS OCCUR IN MID-SCREEN AS WELL AS AT THE EDGES. TRY INSERTING A REVCNT AFTER THE 'CLEAR' COMMAND WITH A SMALL ODD COUNT AND NOTICE THE DIFFERENCE WHEN YOU HIT THE RETURN KEY. CONTINUE TO EXPERIMENT.

MACROS

MACROS ARE THE THIRD MAJOR CATEGORY OF STATEMENTS IN THE CEEMAC LANGUAGE. THE OTHERS WERE 'CONTROL' STATEMENTS AND 'TARGET' STATEMENTS.

NEARLY 30 MACROS ARE PRESENTLY SUPPORTED RANGING FROM SCREEN COMMANDS TO DEBUGGING TOOLS. THE TERMS 'MACRO' AND 'COMMAND' ARE OCCASIONALLY USED INTERCHANGEABLY HEREIN.

EACH MACRO CAN HAVE 2 (OR LESS) PARAMETERS (PARAM1 AND PARAM2). ALSO. CERTAIN VARIABLES MAY HAVE TO BE PRESET.

SOME MACROS ARE 'APPLE ONLY' OWING TO PECULIARITIES UNIQUE TO THE APPLE II COMPUTER. THIS DISTINCTION IS MADE BECAUSE CEEMAC ATTEMPTS TO DISTINGUISH SUCH NON-UNIVERSAL CHARACTERISTICS TO INSURE COMPATIBILITY WITH MORE 'NORMAL' SCREENS.

A DETAILED LISTING OF ALL SUPPORTED MACROS FOLLOWS. OCCASIONALY. THE USE OF A MACRO WILL INVOLVE A DEEPER UNDERSTANDING THAN CAN BE HANDLED HERE AND SEPARATE DOCUMENTATION IS DEVOTED TO ITS EXPLANATION.

--)CLEAR (THE HIRES SCREEN)

PRESET: NONE
PARAM1: BACKGROUND COLOR
PARAM2: UNDINED (RESERVED FOR TEXTURE. SATURATION. GREY SCALE. ETC).

--)SETASP (SET SCREEN ASPECT RATIO)

PRESET: NONE
PARAM1: RATIO CODE
        $00= SQUEEZE Y-COORD TO SCREEN
        $01= NO DISTORTION (CLIP TOP/BTM)
        $02= NO DISTORTION (FIT X TO Y)
        $03= REVERSE DISTORTION (SQUEEZE X)
PARAM2: UNDEFINED (LEAVE ZERO)

(SEE 'SCREEN CONCEPTS')

--)PULSE (REVERSE SCREEN COLOR PHASE)

PRESET: SCREEN NOT ALL BLACK
PARAM1: UNDEFINED (LEAVE ZERO)
PARAM2: NUMBER OF REVERSALS

--)ADOT (PLOT A SINGLE POINT)

PRESET: X/Y COORDINATES (SEE PARAM1)
        COLOR= COLOR CODE
PARAM1: COORDINATE SET CODE:
        $00= XY0
        $01= XY1
        $02= XY2
        $03= XY3
       (OTHERS DEFAULT TO XY0)
PARAM2: SYMMETRY CODE

NOTE: NO AUTO-STAK (SEE 'STACKS' CHAPTER)

--)BLINE (DRAW A STRAIGHT LINE)

PRESET: XY1= START POINT
        XY2= END POINT
        COLOR= COLOR CODE
PARAM1: UNDEFINED (LEAVE ZERO)
PARAM2: SYMMETRY

```
NOTE: AUTO-STAK IN EFFECT

-->ABOX (DRAW A 4-SIDED FIGURE)

PRESET: COLOR= COLOR CODE
        XY1= UPPER LEFT POINT
        XY2= LOWER RIGHT POINT
       (NEXT ONLY FOR QUADRANGLES)
        XY0= UPPER RIGHT POINT
        XY3= LOWER LEFT POINT
PARAM1: 'DRAW' CONTROL CODE
        $00= RECTANGLE, FILLED IN
         (VERT SWEEP, HORIZ LINES)
        $01= RECTANGLE, FILLED IN
         (HORIZ SWEEP, VERT LINES)
        $02= UNDEFINED (RESERVED)
       (UNDEFINED 'PLUS' DEFAULTS TO $00)
        $80= RECTANGLE, OUTLINE
        $81= RECTANGLE, OUTLINE (YES, SAME AS $80)
        $82= QUADRANGLE, OUTLINE
       (NOTE: QUADRANGLES REQUIRE XY0 AND XY3  PRESETS)
       (UNDEFINED 'MINUS' DEFAULTS TO $80)
PARAM2: SYMMETRY CODE

NOTE: NO AUTO-STAK

-->SPLINE

PRESET: COLOR= COLOR CODE
        XY1= START POINT
        XY2= END POINT
        FORCE1 (OR XY0)
        FORCE2 (OR XY3)
        DENSTY (OPTIONAL)
PARAM1: CURVE OPTION BYTE (MULTIPLE CONDITIONS MUST BE 'DRA'D)
        $00= RELATIVE 'FORCES'
        $01= SOFTER CURVES (IF GHOSTPOINTS USED)
        $08= ABSOLUTE 'FORCES'
        $10= XY3 'GHOSTPOINT' FOR XY2
        $40= XY0 'GHOSTPOINT' FOR XY1
        $50= 'GHOSTPOINTS' FOR XY1 AND XY2
PARAM2: SYMMETRY CODE

NOTE: AUTO-STAK IN EFFECT

 (SEE 'SPLINES' AND 'STACKS' CHAPTERS)

-->SHAPE (DRAW A SHAPE)

PRESET: XY1= ORIGIN POINT
PARAM1: SHAPE#
PARAM2: SYMMETRY CODE

NOTE:  UNDEFINED SHAPE NUMBERS DEFAULT TO A RANDOM SELECTION.
NOTE:  NO AUTO-STAK

  (SEE 'SHAPES' CHAPTER)

-->SSIZE (CHANGE THE SHAPE DOWNSCALER)

PARAM1: SHAPE#
PARAM2: SIZE CODE (0-7)

NOTE: IF THE GIVEN SHAPE# DOESN'T EXIST, THIS STATEMENT WILL BE IGNORED.
NOTE: SIZE CODES ARE FORCED MOD-8

 (SEE 'SHAPES' CHAPTER)
```

--)ERASE (ERASE THE NTH OLDEST BLINE)

PRESET: NONE
PARAM1: 'AGE' OF BLINE TO ERASE (0-$3F)
PARAM2: UNDEFINED

(SEE 'STACKS' CHAPTER)

--)SPRASE (ERASE THE NTH OLDEST SPLINE)

PRESET: NONE
PARAM1: 'AGE' OF SPLINE TO ERASE (0-#1F)
PARAM2: UNDEFINED

(SEE 'STACKS' CHAPTER)

--)EXTEND (..THE LAST SPLINE)

PRESET: SPLINE PREVIOUSLY DRAWN
PARAM1: NEXT GHOSTPOINT, X-COORD
PARAM2: NEXT GHOSTPOINT, Y-COORD

NOTE: XY-COORDINATE SET ADVANCES:
XY0(--XY1, XY1(--XY2, XY2(--XY3, XY3(--PARAM1/2

(SEE 'SPLINES' CHAPTER)

--)STAR (PLOT A RANDOM POINT ON SCREEN)

PRESET: NONE
PARAM1: UNDEFINED
PARAM2: COLOR (BLUE IF $00)

--)SKY (FILL SCREEN WITH STARS)

PRESET: NONE
PARAM1: STAR COUNT
PARAM2: COLOR (BLUE IF $00)

--)GRID (DRAW SIMPLE GRID ON SCREEN)

PRESET: (SEE NOTE)
PARAM1: X-INCREMENT
PARAM2: Y-INCREMENT

NOTE: NO BORDER
NOTE: ASSUMPTION IS THAT THIS MACRO IS GIVEN NEAR THE START OF SCORE AND
THEREFORE:
1- XY1 & XY2 ARE AVAILABLE
2- OK TO CLEAR SCREEN
3- COLOR IS PRESET (NORMALLY WHITE)

--)XYSCAL (SET XSCALE AND YSCALE)

PRESET: NONE
PARAM1: SCALE FOR X
PARAM2: SCALE FOR Y

NOTE: PARAMS OF 0 DEFAULT TO 1

--)ISCALE (INCREMENT X/Y SCALES)

PRESET: (X/Y SCALES)
PARAM1: X-SCALE INCREMENT VALUE

PARAM2: Y-SCALE INCREMENT VALUE

NOTE: NO DEFAULT CONVERSION (0=0)

--)DSCALE (DECREMENT X/Y SCALES)

PRESET: (X/Y SCALES)
PARAM1: X-SCALE DECREMENT VALUE
PARAM2: Y-SCALE DECREMENT VALUE

NOTE: NO DEFAULT CONVERSION (0=0)

--)SWAP (EXCHANGE TWO VARIABLES)

PRESET: (TWO VARIABLES PRESET)
PARAM1: 1ST VARIABLE
PARAM2: 2ND VARIABLE

NOTE: PARAMS CANNOT BE LITERALS.

--)NUPOOL (RESET 'POOL' POINTS= RANDOM)

PRESET: NONE
PARAM1: STARTING POOL# (P?)
        $00= P1 (DEFAULT)
PARAM2: ENDING POOL# (P?)
        $00= PF (DEFAULT)

 (SEE 'VARIABLES' CHAPTER)

--)PUSH (SAVE A VARIABLE & RESET IT)

PRESET: (VARIABLE PRESET)
PARAM1: VARIABLE TO SAVE AND RESET
PARAM2: NEW VALUE FOR VARIABLE

--)PULL (RESTORE SAVED VARIABLE VALUE)

PRESET: VARIABLE WAS SAVED WITH 'PUSH'
PARAM1: VARIABLE 'TARGET'
PARAM2: STAK CONTROL CODE:-
        $00= NO ACTION

NOTE: A NON-ZERO PARAM2 IS TAKEN TO MEAN 'CLEAR THIS VARIABLE'S STACKED VALUES'.

SPECIAL NOTE: AFTER RELEASE 1.1, THE USE OF PARAM2 WAS CHANGED SO COMPOSERS ARE ADVISED))NOT((TO USE PARAM2 IN THE MANNER DESCRIBED SINCE IT WONT OPERATE THIS WAY IN THE FUTURE. FOR NOW, SIMPLY LEAVE PARAM2 ZERO.

--)SPEED (SET SPEED; OVERRIDE PDL0)

PRESET: NONE
PARAM1: UNDEFINED
PARAM2: SPEED VALUE
        $FF = FASTEST
        $FE = NEXT FASTEST
              :
              :
        $01 = SLOWEST
       ($00 DEFAULTS TO $FF)

--)WAIT (HALT EXECUTION FOR A WHILE)

PRESET: NONE

PARAM1: FULL SECONDS WAIT
PARAM2: 256TH SECONDS WAIT

--)ANOTE (SOUND A NOTE ON THE APPLE SPEAKER AND 'CASSETTE OUT' PORT)

PRESET: NONE
PARAM1: PITCH
        $00= SILENCE FOR DURATION
        $01= 60 HZ (LOWEST TONE)
        $02= 120 HZ
            :
            :
        $FF= 15,300 HZ (HIGHEST TONE)
PARAM2: DURATION
        $00= (SEE NOTE)
        $01= 1/64TH SECOND
            :
        $40= 1 SECOND
            :
        $FF= 3 63/64THS SECONDS

NOTE: PARAMS [0,0] = APPLE 'BELL'
NOTE: XOPT/BIT10 WILL KILL OUTPUT TO THE APPLE SPEAKER.

--)TRACE (DEBUGGING AID)

PRESET: NONE
PARAM1: 1ST VARIABLE
PARAM2: 2ND VARIABLE (OPT)

NOTE1: ')' IS THE TRACE WINDOW SWITCH ON KEYBOARD.
NOTE2: WILL ACCEPT LITERALS IN PARAM1 AND/OR PARAM2 BUT WILL TRACE UNDEFINABLE VARIABLES.
NOTE3: DONT TRACE (OR STEP) THROUGH 'SYSTEM-SUPPORTED' 'AUTO-CHANGE' VARIABLES SUCH AS RANDOM, RND1, RND2... RND7, NXTCOL, RNDPRM, ETC. THIS WILL CAUSE POSSIBLY UNDESIRABLE (AND PERHAPS MISLEADING) CHANGES TO THEM.

--)PAUSE (WAIT TIL RELEASED BY KEYIN)

PRESET: NONE
PARAM1: UNDEFINED (LEAVE ZERO)
PARAM2: UNDEFINED (LEAVE ZERO)

NOTE: ANY KEYIN WILL CONTINUE EXECUTION.

--)STEP (TRACE WITH PAUSE)

PRESET: NONE
PARAM1: 1ST VARIABLE
PARAM2: 2ND VARIABLE (OPT)

NOTE: ANY KEYIN WILL CONTINUE EXECUTION.

SYMMETRY

SYMMETRY IS AMONG THE MOST POWERFUL TOOLS AVAILABLE TO THE CEEMAC COMPOSER. IT REFLECTS THE ESSENTIAL MIRRORING IN OUR LIFE FORMS. WHEN TO USE IT AND WHEN NOT TO IS STRICTLY THE COMPOSER'S CHOICE; NO 'RULES' ARE AS YET APPARENT. SOMETIMES, SCORES THAT SEEM TO BE 'GETTING NOWHERE' SUDDENLY EXPLODE WITH HIDDEN BEAUTY WHEN A SYMMETRY CODE IS ADDED OR ALTERED. CREATIVE USE OF VARIABLES TO DEFINE (AND REDEFINE) THE CODES CAN PRODUCE SOME EXTRAORDINARY EFFECTS.

SYMMETRY IS SUPPORTED IN THE INTERPRETER AT A VERY LOW LEVEL, WHICH IS WHAT ALLOWS THE RELATIVELY HIGH EXECUTION SPEEDS. THE LANGUAGE, HOWEVER, SUPPORTS IT AT THE HIGHEST LEVEL AS PARAM2 IN THE FOLLOWING MACROS:

ADOT - BLINE - SPLINE - ABOX - SHAPE

LIKE MOST CODES IN CEEMAC, THESE WERE SELECTED TO PROVIDE 'POWERS-OF-2' GROUPINGS, ALTHOUGH THIS OBJECTIVE IS NOT TOTALLY ACHIEVED. ONLY CODES $0 THRU $B ARE UNIQUELY SUPPORTED. (CODES $C THRU $F ARE REPEATS).

IT IS ASSUMED THAT VISUAL COMPOSERS WILL QUICKLY PICK UP ON THE TERMINOLOGY USED (WHICH MAY NOT AGREE WITH THE TEXTBOOKS). EXPERIMENTATION WITH A REAL SCORE SHOULD QUICKLY RESOLVE ANY QUESTIONS.

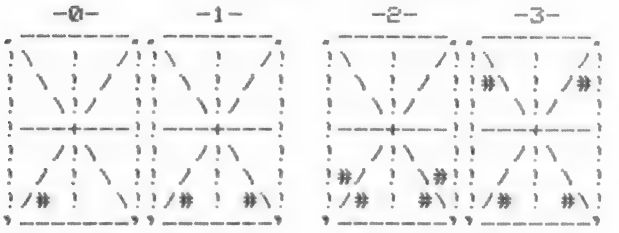
-------(PRIMARILY VERTICAL)---------

CODE-0 = NO SYMMETRY AT ALL

CODE-1 = VERTICAL MIRROR (ONLY)

CODE-2 = VERTICAL MIRROR
+ ADJACENT DIAGONAL

CODE-3 = VERTICAL MIRROR
+ OPPOSITE DIAGONAL



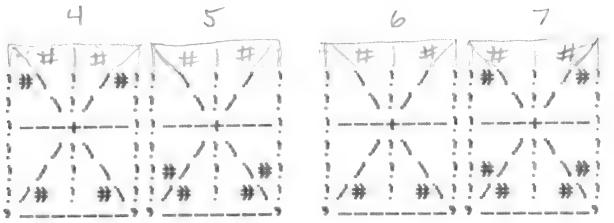
------(VERTICAL + HORIZONTAL)-------

CODE-4 = VERTICAL MIRROR
+ OPPOSITE DIAGONAL
+ HORIZONTAL MIRROR

CODE-5 = VERTICAL MIRROR
+ ADJACENT DIAGONAL
+ HORIZONTAL MIRROR

CODE-6 = VERTICAL MIRROR
+ HORIZONTAL MIRROR

CODE-7 = VERTICAL MIRROR
+ HORIZONTAL MIRROR
+ ADJACENT DIAGONAL
+ OPPOSITE DIAGONAL



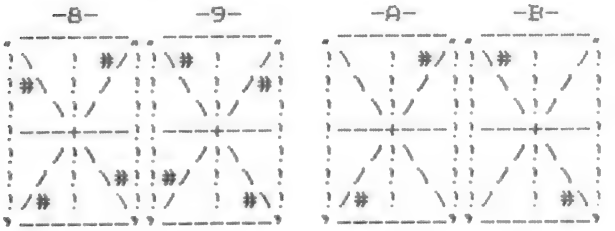
----------(SPIRALS)----------

CODE-8 = 4-POINT SPIRAL LEFT

CODE-9 = 4-POINT SPIRAL RIGHT

CODE-A = 2-POINT SPIRAL RIGHT

CODE-B = 2-POINT SPIRAL LEFT



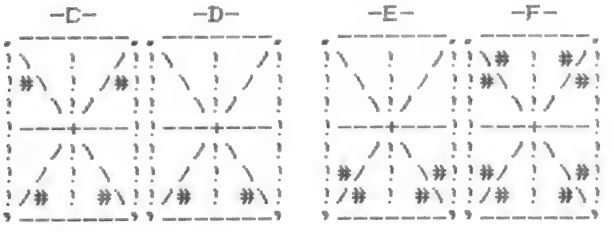
---------(SELECTED REPEATS)---------

CODE-C = SAME AS CODE-3

CODE-D = SAME AS CODE-1

CODE-E = SAME AS CODE-2

CODE-F = SAME AS CODE-7



THE CODES ARE GROUPED AS THEY ARE TO AID THE COMPOSER IN CONFINING THE VARIATIONS WHEN HE WISHES TO DO SO. THE APPLICATION OF BOOLEAN RULES AND/OR RANGE BOUNDING PROVIDES AT LEAST SOME CONTROL. E.G. TO USE THE VF VARIABLE FOR THE CODE BUT RESTRICT IT TO ONLY THE SPIRALS:

```
-->VF = RND2 + 8
-->BLINE [0;VF]
```

HAVING SET UP THE VARIABLE THIS WAY WE COULD THEN SWAP FROM LEFT-HANDLED SPIRALS TO RIGHT-HANDED SPIRALS BY:

```
-->VF = VF EOR 1
```

ANOTHER CONSIDERATION IS ALLOWING A RANGE OF SYMMETRIES WHILE INSURING THAT CODE-0 IS NEVER USED:

```
-->NUPOOL [P1;P7]
     :
     ;
-->BLINE [0;P4]  (OR [..;P3], ETC)
```

IT CAN BE SEEN THAT THE ABOVE IS SIMPLER THAN INSURING NON-ZERO BY 'TEST AND SET' SINCE THE P1-P7 RANGES EXCLUDE THE ZERO POSSIBILITY.

CODES $10 THRU $FF ARE 'MOD'ED $10 (BY 'AND'ING WITH $0F) AND $0 RESULTS ARE FORCED TO $1. AS NEW SYMMETRIES ARE DEFINED, FUTURE EXPANSION OF THE CODING STRUCTURE ABOVE $F WILL OCCUR.

LISTS

THE IS REALLY ABOUT 'INDEXING'. WHEN YOU HAVE A STRING OF VALUES, IT IS HANDY TO BE ABLE TO REFERENCE ANY ONE BY SIMPLY SPECIFYING ITS POSITION IN THE LIST. IN CEEMAC, THIS IS DONE BY PUTTING THE NAME OF THE LIST IN OP1 AND THE 'INDEX' VARIABLE IN OP2 AS IN THIS STATEMENT:

```
-->Y3 = LIST1 , X3
```

LIST1 IS THE NAME OF THE FIRST PAGE IN 'LISTS', A FREESTANDING MODULE CONSISTING OF FOUR PAGES. THE FIRST LIST IN THE MODULE (LIST1) CONSISTS OF 256 POINTS OF A FULL SINE WAVE. LOOK AT THIS EXAMPLE:

```
:     -- DRAW A SINE WAVE --
CLEAR [0;0]
X3 = 0
DO
--->Y3 = LIST1 , X3
   ADOT [3,0]
   X3 = X3 + 1
AGAIN WHILE X3 > 0
: -- END OF SINE WAVE DRAW --
```

THIS SCORE WILL TRACE A SINE WAVE FROM LEFT TO RIGHT ACROSS THE SCREEN. IF YOU INCREMENT X3 BY 2 YOU WILL GET HALF THE POINTS ON THIS WAVE BUT IT WILL TRACE TWICE AS FAST. IF YOU NOW CHANGE PARAM2 OF THE 'ADOT' MACRO TO 1, YOU'LL GET OVERLAPPING SINE WAVES.

LIST2 HAS A 'DAMPING' SINE WAVE. SIMPLY SUBSTITUTE 'LIST2' IN THE ABOVE SCORE TO SEE WHAT IT LOOKS LIKE.

LIST3 HAS 128 POINTS ON A CIRCLE AND LIST4 HAS 128 POINTS ON A SPIRAL. TWO BYTES ARE USED FOR EACH POINT (X AND Y) IN LIST3 AND LIST4. EXAMPLES OF HOW TO USE THEM CAN BE FOUND IN THE 'VARIABLES' CHAPTER.

IN RELEASE 1.1, THE LIST MODULE IS LOCATED AT $5000 AND IS FOUR 'PAGES' LONG, ONE FOR EACH LIST. JUST AS WITH SHAPES, YOU MAY WISH TO DESIGN YOUR OWN FOR USE WITH PARTICULAR SCORES. MIXING VARIOUS LISTS WITH DIFFERENT SCORES CAN CREATE SOME WILD AND UNEXPECTED EFFECTS.

YOU MIGHT EVEN WANT TO CREATE YOU OWN LIBRARY OF LISTS WHICH CAN THEN BE INDIVIDUALLY LOADED TO $5000, $5100, $5200 OR $5300 FOR THE SCORE YOU ARE WORKING ON. NOTE THAT DOS REMEMBERS WHERE YOU BSAVED FROM. IF YOU WANT IT ELSEWHERE, YOU'LL HAVE TO APPEND THE ',A$5X00' PART WHEN BLOADING IT BACK. IT MIGHT PAY TO INCLUDE A DIGIT 1-4 IN YOUR FILE NAME TO HELP YOU REMEMBER WHERE IT'S ORIGINED.

YOU COULD FIND OTHER USES FOR THE INDEXING FUNCTION. OR EVEN OTHER USES FOR THE 'LISTS' MODULE ITSELF.

SPLINES

THIS EXPLANATION IS MEANT TO BE AS NON-TECHNICAL AS POSSIBLE AND STILL INCLUDE A THEORETICAL FOUNDATION FOR THOSE WITH THAT LEVEL OF INTEREST. THE ALGORITHM USED FOR THE SPLINES IN CEEMAC WAS GENEROUSLY SUPPLIED BY MIK JORDAN, PHD. THE CONCEPTUALIZATION THAT FOLLOWS IS ONLY ONE OF SEVERAL POSSIBLE VIEWS OF THE PRINCIPLES OF THIS OBSCURE AND STRIKING GEOMETRIC FIGURE.

VISUALIZE HOLDING A SAW BLADE AT BOTH ENDS. VIEW THE BLADE 'ON EDGE' SINCE WE WILL LIMIT OUR CONCEPTS TO A TWO DIMENSIONAL PLANE. AS YOU WORK YOU WRISTS, THE SEMI-FLEXIBLE METAL BOWS IN OR OUT IN RESPONSE TO THE FORCES YOU APPLY. IF BOTH ENDS ARE FORCED OUTWARD OR BOTH INWARD, A SIMPLE CURVE RESULTS. IF ONE END IS FORCED INWARD WHILE THE OTHER END IS FORCED OUTWARD, THE BLADE DESCRIBES AN 'S' CURVE. HOLDING ONE END FIXED WHILE APPLYING FORCE TO THE OTHER END CREATES AN UNBALANCED 'S' CURVE EFFECT.

UNLIKE THE SAW BLADE, SPLINES IN CEEMAC ARE ELASTIC SO THE END POINTS THEMSELVES DONT MOVE UNDER PRESSURE. BUT VERY MUCH LIKE THE SAW BLADE, SPLINES RESPOND TO 'FORCES'. THE VARIABLE THAT CONTROLS THIS PRESSURE AT THE START POINT (XY1) WE CALL 'FORCE1' WHILE THE END POINT (XY2) WILL RESPOND TO 'FORCE2'. THE COMBINED EFFECT OF THESE VARIABLES IS SUCH THAT A WIDE VARIETY OF COMPLEX CURVES CAN BE GENERATED BY SIMPLY CHANGING THESE FORCE VARIABLES.

TO SEE HOW THIS WORKS AND TO SET A FRAME OF REFERENCE, CEEMAC UTILIZES CLOCKFACE VALUES TO SPECIFY THE FORCES. CUTTING A STANDARD CIRCLULAR CLOCKFACE ACROSS ITS 3 O'CLOCK - 9 O'CLOCK AXIS, WE CAN ATTACH EACH HALF TO THE ENDS OF A VERTICAL SPLINE (OUR SAW BLADE ON EDGE). THE SPLINE IS RELAXED WHEN THE FORCE AT THE TOP END IS COMING FROM 12 O'CLOCK AND THE FORCE AT THE BOTTOM IS COMING FROM 6 O'CLOCK. AT THIS TIME THE SPLINE IS ESSENTIALLY A STRAIGHT LINE.

WE CAN 'APPLY FORCE' TO THE SPLINE THRU THE START POINT (XY1) BY MOVING OUR IMAGINARY 'FORCE LINE' AROUND THE TOP HALF-CLOCKFACE. THIS IS DONE BY CHANGING THE VALUE IN THE FORCE1 VARIABLE. IF WE SET FORCE1 TO 1:00, THE FORCE LINE CAN BE SAID TO 'EXERT PRESSURE' ON THE SPLINE TO BOW OUTWARD TOWARD 7:00. OF COURSE, THIS PRESSURE MUST BE BALANCED AGAINST THE RESISTING END POINT AT XY2 AS WELL AS ANY PRESSURE IT MAY BE RECEIVING FROM FORCE2.

IT CAN BE SEEN THAT FORCE2 CAN EXERT SIMILAR PRESSURE ON THE END POINT (XY2) BY CHANGING IT'S REST POSITION FROM 6:00.

MEANWHILE, BACK AT FORCE1, WE ARE PRESURING THE START POINT (XY1) SINCE WE'RE 'COMING FROM' 1:00. IF WE CHANGE FORCE1 TO 1:30, THE PRESSURE ON THE XY1 POINT INCREASES SIGNIFICANTLY. AS WE MOVE FURTHER TOWARD 3:00, THE PRESSURE BECOMES ALMOST UNBEARABLE AND THE SPLINE BOWS OUTWARD TOWARD 9:00 IN AN EXTREMELY DISTENDED MANNER (ALSO WITH A QUITE STRIKING EFFECT). BY THE TIME FORCE1 REACHES 3:00 THE SPLINE TRIES TO DESCRIBE AN 'S' CURVE DISTENDED TO THE POINT WHERE IT IS PARALLEL TO ITSELF. A MATHEMATICAL DILEMMA RESULTS AND NO CURVE CAN BE DRAWN.

EVEN AT 2:00, THE DISTORTION IS CONSIDERABLE AND THE SPLINE STARTS TO LOSE DEFINITION. CONCEPTUALLY, WHAT 2:00 IS TO FORCE1 AND ITS EFFECT ON XY1, 8:00 IS TO FORCE2'S EFFECT ON XY2. THE COMBINATION OF THESE TWO FORCES GIVES THE SPLINE ITS PARTICULAR CURVATURE. INCIDENTLY, IN RELEASE 1.1, CEEMAC CAN'T DRAW THE ABSOLUTE VERTICAL RELAXED SPLINE. A LATER RELEASE WILL FIX THIS.

MATHEMATICALLY, THE 'FORCE LINES' WE'VE BEEN TALKING ABOUT ARE SLOPES WHICH HAVE NO DIRECTIONALITY. IN PRACTICE, A FORCE VALUE AT 1:00 IS IDENTICAL TO THAT AT 7:00. FURTHUR EXPLANATION OF THIS POINT MIGHT ONLY CONFUSE. THE EFFECTS CAN BE SEEN BY SOME SIMPLE SCORES WHERE THE FORCES ARE INCREMENTED AROUND THE CLOCKFACE, PREFEREABLY MONITORING THE FORCE1 AND FORCE2 VALUES WITH THE TRACE OR STEP MACROS IN THE LOOP.

CEEMAC CARRIES CLOCKFACE VALUES INTERNALLY AS 'N/256THS' OF A CIRCLE. IN DOING SO, SOME POSITION RESOLUTION IS LOST. YOU MIGHT SPECIFY A POSITION OF 1:00 AND CEEMAC WILL CONVERT IT TO 12:59, THE CLOSEST POINT IT CAN REMEMBER. MICKEY MOUSE? YES. EFFICIENT? ALSO, YES. ACCEPTABLE? YOU DECIDE.

SPLINES ARE NOT DRAWN WITH SOLID LINES BUT RATHER A SERIES OF DESCRETE DOTS WHERE THE SPACING VARIES GRADUALLY WITHIN THE LINE. CEEMAC WILL CALCULATE THE

POINT COUNT ITSELF AS A DEFAULT CONDITION IF THE VARIABLE 'DENSTY' IS NOT SUPPLIED IN THE SCORE. COMPOSERS CAN SET THIS VARIABLE IN THE NORMAL MANNER AND THEREBY INCREASE OR DECREASE THE SOLIDITY OF THE LINE. AGAIN, EXPERIMENT.

JUST AS WITH BLINES, DOTS, BOXES AND SHAPES, PARAM2 SPECIFIES THE SYMMETRY CODE. PARAM1 PROVIDES SOME INTERESTING OPTIONS FOR DRAWING SPLINES WHICH ARE NOW EXAMINED. THE SPECIFIC CODES FOR THESE VARIATIONS CAN BE FOUND IN THE 'MACROS' CHAPTER AND ONLY THEIR FUNCTIONS ARE COVERED HERE.

THE FORCES PREVIOUSLY DESCRIBED WORK ON A 'RELATIVE' BASIS. THAT IS TO SAY, THE PARTICULAR LOCATION OF THE END POINTS ON THE SCREEN DO NOT EFFECT THE CURVE OF THE SPLINE. THEORETICALLY, YOU SHOULD BE ABLE TO MOVE THE SPLINE ABOUT THE SCREEN WITHOUT CHANGING ITS SHAPE. IN PRACTICE THIS IS DIFFICULT BECAUSE THE ACTUAL DISTANCE BETWEEN END POINTS WILL SOMEWHAT EFFECT THE RELATIVE DISTENTION.

A BIT IN PARAM1, IF SET, MAKES THE FORCE VARIABLES WORK ON AN 'ABSOLUTE' BASIS. NOW THE END POINT POSITIONS OF XY1 AND XY2 BECOME ACTIVE PARTICIPANTS IN THIS VISUAL SYMPHONY. 'TAMING' THIS EFFECT HAS YET TO BE ACCOMPLISHED BUT IT IS BEAUTIFUL (AND AVAILABLE).

AN ALTERNATIVE TO THE 'FORCES SET AT CLOCKFACE POINTS' APPROACH IS ALSO AVAILABLE. ITS CALLED 'GHOSTPOINTS'. XY0 IS THE GHOSTPOINT FOR XY1 AND XY3 IS GHOSTPOINT FOR XY2. IF THE APPROPRIATE BIT IN PARAM1 IS SET, CEEMAC WILL KNOW THE 'FORCE LINE' BY CALCULATING THE SLOPE BETWEEN XY0 AND XY1. ANOTHER BIT DOES THE SAME FOR THE OTHER END OF THE SPLINE WITH XY3 AND XY2. ONE END OF THE SPLINE CAN BE 'DRIVEN' BY A FORCE VALUE AND THE OTHER BY A GHOSTPOINT (ALTHOUGH ITS NOT CLEAR WHY YOU WOULD WANT TO).

DEPENDING UPON THE EFFECT DESIRED, GHOSTPOINTS MAY BE MORE USEFUL. THERE IS A MACRO CALLED 'EXTEND' WHICH IS DESIGNED TO SUPPORT SERPENTINING OF SPLINES AND PRESUMES THAT THIS OPTION IS BEING USED.

STILL ANOTHER BIT IN PARAM1 WILL ALLOW FOR 'SOFTER' CURVES WHEN USING GHOST POINTS. THIS IS LARGELY UNEXPLORED TERRITORY.

AN UNRESOLVED TECHNICAL PROBLEM CAUSES A NON-DRAW WHEN X1=X2. UNTIL RESOLVED, AVOID, IF POSSIBLE, THE OCCURANCE.

SCALING? ROTATION? INCLUDE IN SHAPES? NONE OF THESE ARE PRESENTLY SUPPORTED FOR SPLINES. WILL THEY EVER BE? PROBABLY, AS THERE APPEARS NO CONCEPTUAL REASON WHY NOT. HOW SOON, IS ANOTHER QUESTION.

SHAPES

THE APPLE SHAPE SYSTEM HAS BEEN ABANDONED FOR MANY REASONS WHICH WILL NOT BE LISTED HERE. IN ITS PLACE IS A MORE FLEXIBLE, EXPANDABLE SYSTEM BUT PRESENTLY MISSING DOS TRANSPARENCY.

SHAPE TABLES:

A SHAPE TABLE CONSISTS OF A SERIES OF SHAPE 'ENTRIES' WHICH CAN BE PUT TOGETHER FROM INDEPENDENTLY 'BSAVED' SHAPE MODULES. THE SHAPE TABLE 'HEADER' IS THE FIRST 8 BYTES OF THE TABLE, DEFINED AS FOLLOWS:

```
BYTE-0: TABLE# (1-$FF)
BYTE-1: UNDEFINED
BYTE-2: SHAPE COUNT (0-$FF)
BYTE-3 THRU BYTE-7: UNDEFINED
```

NOTE: ALL UNDEFINED FIELDS SHOULD BE ZERO.

NOTE: AT PRESENT, ALL OF THE ABOVE IS IGNORED BY THE INTERPRETER; IT SIMPLY BYPASSES THE FIRST 8 BYTES. THIS 'HEADER' RECORD WILL BE MORE USEFUL IN LIBRARIAN OPERATIONS AND WITH TABLE BUILDING ROUTINES LATER. INDEPENDENTLY WRITTEN SHAPE TABLE GENERATORS SHOULD SUPPORT THESE FIELDS FOR FUTURE UPWARD COMPATIABILITY.

SHAPE ENTRIES:

EACH SHAPE IN A SHAPE TABLE IS DEFINED AS A 'SHAPE ENTRY' AND HAS ITS OWN 8-BYTE 'HEADER', DEFINED AS FOLLOWS:

```
BYTE-0: SHAPE# (1-$FE)
BYTE-1: (3 FIELDS)
  FIELD-1, MASK=$80: UDEFINED
  FIELD-2, MASK=$70: SIZE (0-7)
  FIELD-3, MASK=$0F: STARTING FORMAT (0-$F)
BYTE-2: (2 FIELDS)
  FIELD-1, MASK=$F8: UNDEFINED
  FIELD-2, MASK=$07: LENGTH-HI
BYTE-3:    MASK=$FF: LENGTH-LO
BYTE-4: SHAPE WIDTH
BYTE-5: SHAPE HEIGHT
BYTE-6: PIXEL COUNT (MOD-16)
BYTE-7: UNDEFINED
```

NOTES ON SHAPE ENTRY HEADER FIELDS:

ALL UNDEFINED FIELDS SHOULD BE LEFT ZERO.

SHAPE#0 IS ALWAYS CONVERTED TO SHAPE#1 BY THE INTERPRETER.

SHAPE#$FF IS RESERVED FOR END-OF-TABLE USE.

IF THE SHAPE# SOUGHT DOESN'T EXIST IN THE TABLE, THE INTERPRETER WILL SELECT A SHAPE BASED ON AN OBSCURE TECHNIQUE WHERE THE SELECTION MADE IS UNPREDICTABLE BUT BIASED TOWARD THE END OF THE TABLE.

'SIZE' IS AN INTERNAL DOWNSCALER. AFTER THE INTERPRETER HAS LOCATED THE SHAPE ENTRY IT REDUCES IT BY THE POWERS-OF-2 VALUE FOUND IN THIS 3-BIT CONTROL FIELD. THIS PERMITS CREATION OF SHAPES ON A LARGE ENUF SCALE TO INSURE SUFFICIENT DETAIL. IT ALSO LETS US ADJUST A SHAPE TO SCREENING NEEDS WITHOUT MODIFYING THE LINE SEGMENT VALUES. COMBINED WITH THE XSCALE AND YSCALE VARIABLES (WHICH ARE UPSCALERS), ALMOST ANY VIEWABLE SIZE CAN BE HAD FROM ANY EXISTING SHAPE DEFINITION. THIS FIELD CAN BE ALTERED FOR SCORE EXECUTION BY THE MACRO 'SSIZE'.

THE 'FORMAT' (4-BIT) FIELD ALLOWS US TO UTILIZE A VARIETY OF TECHNIQUES TO ENCODE THE ACTUAL LINE SEGMENTS. MOST CODES ARE UNASSIGNED OR UNDEFINED AT PRESENT:

CODE-0: INVALID
CODE-1: 8-BIT COORDINATES
(SCREEN ABSOLUTE)
CODE-2: 8-BIT OFFSETS
CODE-3: UNDEFINED
CODE-4: 4-BIT OFFSETS
CODE-5->F: UNDEFINED

THESE CODES DEFINE THE MEANING AND STRUCTURE OF THE LINE SEGMENT VALUES WHICH FOLLOW THE HEADER RECORD OF THE SHAPE ENTRY. THEY ARE DISCUSSED IN DETAIL LATER. THE CODE APPEARING IN THE HEADER RECORD IS JUST THE STARTING FORMAT CODE AND CAN BE CHANGED DURING LINE SEGMENT PROCESSSING BY AN 'ESCAPE' SEQUENCE (SEE LATER).

FUTURE POSSIBLE FORMAT CODES:

BLOCK SHAPES
ARCS
SPLINES
12-BIT COORDINATES
12-BIT OFFSETS
DISTANCE/BEARING
APPLE SHAPES (I HOPE NEVER)

THE LENGTH FIELD IS 11 BITS LONG AND SUPPORTS SHAPES REQUIRING UP TO 2048 BYTES.

THE WIDTH, HEIGHT AND PIXEL COUNT FIELDS, WHILE DEFINED, ARE PRESENTLY UNUSED. EVENTUALLY, THEY WILL PROVIDE INFORMATION TO THE INTERPRETER FOR TIMING AND SCREEN DENSITY ANALYSIS. INDEPENDENTLY WRITTEN SHAPE GENERATORS SHOULD SUPPLY THIS DATA FOR UPWARD COMPATABILITY.

LINE SEGMENTS:

'OFFSET' VALUES DEFINE HOW FAR FROM THE LAST POINT TO THE NEXT POINT. THEY ARE SIGNED VALUES AND THUS INDICATE DIRECTION AS WELL AS DISTANCE. THE 8-BIT OFFSET FORMAT REQUIRES 2 BYTES PER POINT. THE 4-BIT OFFSET FORMAT ONLY NEEDS ONE BYTE WITH EACH HEX DIGIT BEING A SIGNED OFFSET OF +OR- 7 PIXELS.

'COORDINATE' VALUES DEFINE AN X/Y LOCATION ON THE 'UNIT' SCREEN (256 X 256) AND ARE UNSIGNED. SCREEN ORIGIN IS 0,0 AT LOWER LEFT. THIS FORMAT NEEDS 2 BYTES FOR EACH X/Y POINT.

ESCAPE SEQUENCES MUST BE BURIED WITHIN THESE ENCODING SCHEMES FOR VARIOUS PURPOSES. THE TECHNIQUE USED IS DISCUSSED HERE WHILE THE ESCAPE CODES THEMSELVES, ARE EXAMINED FURTHER ON.

AN $80 APPEARING IN THE FIRST BYTE OF A LINE SEGMENT MEANS 'ESCAPE'. THIS $80 BYTE IS CALLED THE 'ESCAPE DETECTOR BYTE'. WHEN ENCOUNTERED, THE INTERPRETER WILL USUALLY FIND THE ESCAPE CODE ITSELF IN THE FOLLOWING BYTE. THIS TECHNIQUE IS WORKABLE ONLY IF AN $80 CAN NEVER MEAN A REAL VALUE. FOR 'OFFSET' FORMATS, THIS IS EASY SINCE THEY ARE SIGNED VALUES AND AVOIDING THE USE OF '-0' FOR THE X-OFFSET DOES THE TRICK. 'COORDINATE' VALUES ARE ANOTHER MATTER. WHEN THE X COORDINATE IS $80 (SCREEN CENTER), THEN IT MUST BE))DUPLICATED((IN THE NEXT BYTE FOLLOWED BY THE Y COORDINATE. THIS LETS THE INTERPRETER ESCAPE OUT BUT IMMEDIATELY RETURN TO COORDINATE MODE.

ESCAPE CODES

THE ESCAPE CODE BYTE FOLLOWS THE ESCAPE DETECTOR BYTE ($80):

CODE	USE
STRUCTURE CODES:	
00	NULL CODE (IGNORE)
80	X COORDINATE IS $80
EE	CODE IS IN NEXT BYTE
FE	END OF SHAPE ENTRY
FF	END OF SHAPE GROUP
('SHAPE GROUP' NOT YET DEFINED)	

'GOOD TIL ALTERED' CODES:
  01-0F CHANGE FORMAT (CODE= LO NIB)
     21 CHANGE MODE (=POSN ONLY)
     22 CHANGE MODE (=PLOT DOTS)
     24 CHANGE MODE (=DRAW NEXT LINE)
     26 CHANGE MODE (=DRAW RAY LINES)
'NEXT POINT ONLY' CODES:
  11-1F CHANGE FORMAT (CODE= LO NIB)
     31 CHANGE MODE (=POSN ONLY)
     32 CHANGE MODE (=PLOT DOTS)
     34 CHANGE MODE (=DRAW NEXT LINE)
     36 CHANGE MODE (=DRAW RAY LINES)

ESCAPE SEQUENCES WITH INVALID CODES ARE IGNORED.

THE 'NULL CODE' CAN BE USED TO PATCH OUT AN EXISTING ESCAPE SEQUENCE OR CREATE A SEQUENCE AT SHAPE GENERATION TIME WITHOUT DEFINING IT.

BY RESERVING THE $EE VALUE NOW AS A SORT OF 'TRAPDOOR', THE SYSTEM INSURES ITS FUTURE EXPANDABILITY, ALTHOUGH ALL KNOWN ESCAPE FUNCTIONS SEEM TO FIT WITHIN THE CURRENTLY AVAILABLE STRUCTURE.

A SHAPE ENTRY MUST TERMINATE WITH AN ESCAPE SEQUENCE OF $80-$FE. IT'S SIMPLY THE RULE.

END OF 'SHAPE GROUP' IS NOT FURTHER DEFINED AS OF NOW. IT WILL PROBABLY BE USED FOR 'SUPER SHAPES' COMPOSED OF MULTIPLE RELATED ENTRIES. FOUR-LIMBED 'CREATURE' SHAPES COME TO MIND.

'GOOD TIL ALTERED' CODES ARE THE NORMAL ONES USED TO CHANGE FORMAT OR 'MODE', WHICH WILL THEN NOT CHANGE AGAIN UNTIL THE NEXT ESCAPE SEQUENCE IS ENCOUNTERED.

'NEXT POINT ONLY' CODES PROVIDE A METHOD OF CHANGING FORMAT OR MODE FOR THE NEXT X/Y SET ONLY. WHILE THIS REDUCES EXECUTION TIME SLIGHTLY, ITS MAIN PURPOSE IS TO SAVE SPACE IN THE SHAPE ENTRY. AN OBVIOUS USE IS WHEN CONTINUOUS DRAWING ENDS AND A NEW START POINT IS TO BE SET (THAT IS, A 'GAP' IN THE DRAWING). ANOTHER USE IS FOR INSERTING A LONG (8-BIT) OFFSET WITHIN A SHAPE THAT PRIMARILY USES 4-BIT OFFSETS, AS WITH CHARACTER SETS.

CHANGE-OF-FORMAT IS ACCOMPLISHED BY A HIGH NIBBLE OF $0 OR $1 AND THE LOW NIBBLE HAVING THE NEW FORMAT CODE ITSELF.

CHANGE-OF-MODE DEPENDS UPON KNOWING WHAT IS MEANT BY 'MODE'. TWO POINT MODES ARE SUPPORTED. 'POSN' SIMPLY UPDATES THE WORKING START COORDINATE WHILE 'PLOT' DOES THIS AND THEN PLOTS THAT POINT ON THE SCREEN.

TWO LINE MODES ARE ALSO SUPPORTED. CONTINUOUS LINE DRAWING (SERPENTINE) IS THE MOST NORMAL. 'RAY' LINES DIFFER IN THAT THE LATEST LINE IS DRAWN FROM THE LAST DEFINED START POINT RATHER THAN THE LAST DEFINED END POINT.

SCREEN POSITIONING OF SHAPES:

A SHAPE IS POSITIONED ON THE UNIT SCREEN AT A POINT DEFINED BY X1/Y1. THIS COORDINATE SET IS DETERMINED DYNAMICLY DURING SCORE EXECUTION AND IS FUNCTIONALLY SEPARATE FROM THE CEEMAC SHAPE SUB-SYSTEM ITSELF.

THE STARTING MODE OF ANY SHAPE SHOULD USUALLY BE CODE-31. THIS WORKS WELL WHEN THE SHAPE IS TO BE CENTERED AT THE X1/Y1 POSITION. GIVEN THIS, IT IS REASONABLE TO ASSUME THAT THE FIRST THING TO DO IS JUMP TO A POINT WHERE DRAWING CAN COMMENCE. A MATCHING DEFAULT 'RETURN TO' CODE-24 THEN STARTS LINE DRAWING AT THAT POINT.

DEFEAT THIS ARRANGEMENT WITH YOUR OWN ESCAPE SEQUENCE AT THE START, IF YOU CHOOSE.

SCALING AND ROTATION:

CEEMAC SHAPES ARE '2D' PLANES. THEY ARE PRESENTED 'HEADON' AND CAN BE ENLARGED (UPSCALED) BY THE USE OF THE XSCALE AND YSCALE VARIABLES. THESE

SHAPES CAN ALSO BE ROTATED ABOUT THEIR LOGICAL CENTER (THE 'Z-AXIS') BY USE OF THE 'ROTATE' VARIABLE. FUTURE RELEASES OF CEEMAC WILL SUPPORT ROTATION ABOUT THE X-AXIS AND Y-AXIS AS WELL.

CLIPPING:

CEEMAC WILL 'CLIP' ANY LINE SEGMENTS WHICH LOGICALLY MAP OFF THE SCREEN. DEPENDING UPON THE SCALING OR ROTATION, A SHAPE CAN BE 'TOTALLY CLIPPED' AND ONLY BEGIN TO APPEAR AS THE SCALING IS REDUCED.

CONCLUSION:

THIS SHAPE DEFINITION SYSTEM AND ITS SUPPORT IN CEEMAC WILL CONTINUE TO UNDERGO EXPANSION AND MODIFICATION AS TIME GOES BY. CONSIDERABLE EFFORT HAS BEEN EXPENDED TO PROVIDE A FLEXIBLE GROWTH PATH IN THE HOPE THAT SHAPES, ONCE DEFINED, WILL BE UPWARD COMPATIBLE. HOWEVER, THIS CANNOT BE ABSOLUTELY GUARANTEED.

IT IS RECOMMENDED THAT ALL UNDEFINED BITS AND BYTES BE FORCED TO ZERO TO INSURE AT LEAST THIS LEVEL OF CONTINUITY. ONE POSSIBLE UPGRADE PATH WILL INVOLVE 'AUTO CONVERSION' OF OLDER SHAPE ENTRIES PROVIDING THAT THE 'RELEASE NUMBER' IS AVAILABLE. NO ASSIGNMENT IN EITHER THE SHAPE TABLE HEADER OR THE SHAPE ENTRY HEADER HAS BEEN MADE FOR THIS PURPOSE AS YET. ACCORDINGLY, IT WOULD APPEAR THAT THIS IS RELEASE-0, SHAPEWISE.

IT MUST ALSO BE APPARENT THAT SHAPES DEFINED ACCORDING TO THIS SYSTEM ARE NOT LIMITED TO USE IN THE CEEMAC INTERPRETER. INDEPENDENTLY WRITTEN ANIMATION PACKAGES, FOR EXAMPLE, MAY FIND THE STANDARDS SET FORTH HERE TO BE A DESIRABLE ALTERNATIVE TO, SAY, APPLE SHAPES.

MUSIC

THERE ARE TWO METHODS BY WHICH CEEMAC CAN 'GET IN TOUCH' WITH THE CURRENT MUSIC ENVIRONMENT. NEITHER OF THESE METHODS, HOWEVER, IS TOTALLY SATISFACTORY AT THIS TIME. THE TECHNIQUES USED AND THE TRADEOFFS INVOLVED SHOULD BE UNDERSTOOD TO AVOID UNREALISTIC EXPECTATIONS.

METHOD-1; 'KEYED' INPUT:

THIS INVOLVES THE CONCEPT OF USING THE SYSTEM AS A 'VISUAL INSTRUMENT' CAPABLE OF BEING 'PLAYED'. THE PRESENT VERSION IS LIMITED IN THIS REGARD. THERE ARE THREE 'LEVELS' OF SUCH PERFORMANCE ANY OR ALL OF WHICH CAN BE COMBINED AT THE OPTION OF THE PLAYER/COMPOSER.

THE FIRST LEVEL INVOLVES ONLY THE USE OF THE PADDLES & BUTTONS. THE SCORE MIGHT HAVE PRESUMED SUCH A PERFORMANCE BY RELEASING PDL0 FROM ITS NORMAL ROLE OF SPEED CONTROL. PADDLE READINGS CAN BE USED LIKE ANY OTHER VARIABLES. BUTTON READINGS CAN HALT EXECUTION LOOPS AND RESTART SEQUENCES.

THE SECOND LEVEL INVOLVES PLAYING THE 'DRUMS', KEYS 1-5. WHENEVER DEPRESSED, THESE KEYS CAUSE THE CORRESPONDING VARIABLE (DRUM1 - 5) TO BE 'FLIPPED' (OFF=0; ON=NON-ZERO). BY HAVING THE SCORE TEST FOR SUCH CHANGES AND ALTER EXECUTION ACCORDINGLY, THE PERFORMER CAN KEEP TIME WITH THE MUSIC IN ANY MANNER THAT HE AND THE COMPOSER WISH. SCORES THAT UTILIZE THIS ABILITY MIGHT BE 'PLAYABLE' TO A WIDE VARIETY OF MUSIC WITHOUT MODIFICATION.

THE THIRD LEVEL IS ACTUALLY AN EXTENSION OF THE SECOND. AN UNUSUAL INTERNAL PROCEDURE PUTS ANY INVALID KEYIN INTO A VARIABLE CALLED 'KEY' WHICH CAN BE READ BY THE SCORE. THIS PERMITS THE COMPOSER TO UTILIZE ANY KEY NOT DEDICATED TO SYSTEM USE AS AN INPUT SIMPLY BY TESTING 'KEY' FOR HIS CHOSEN ASCII EQUIVALENT. THE SCORE MUST CLEAR THIS VARIABLE AFTER DETECTION SINCE THE INTERPRETER DOES NOT SUPPORT 'DESTRUCTIVE READINGS' OF VARIABLES. SILENCING THE 'CLICK' FROM INVALID KEYINS CAN BE DONE FROM THE KEYBOARD WITH A '=' KEYIN OR FIXED IN THE SCORE WITH THE FOLLOWING STATEMENT:

```
-->XOPT = $20
```

METHOD-2; 'LIVE' INPUT:

THE 'CASSETTE-IN' PORT ON YOUR APPLE (IF IT HAS ONE) WILL READ AN ANALOG (MUSIC) INPUT JUST AS IT WILL READ A PROGRAM THAT HAS BEEN SAVED AS A BIT STREAM, THE PRE-DISK PROGRAM SAVING METHOD. HOWEVER, SINCE THE ORIGINAL INTENT OF THIS PORT WAS TO READ BACK SAVED PROGRAMS, THE ENCODING AND DECODING TECHNIQUES EMPLOYED WERE OPTIMIZED FOR DIGITAL DATA. THIS INVOLVED MAKING NICE, NEAT SQUARE WAVES TO OPTIMIZE LATER READING OF THE ONES AND ZEROS AS ORIGINALLY STORED.

ALLOWING FOR SOME CRUDENESS, IT IS POSSIBLE TO 'READ' THE PITCH OF THIS INPUT PORT WHEN IT IS RECEIVING MUSIC INSTEAD OF A PREVIOUSLY STORED PROGRAM. A MAJOR PROBLEM IS THAT THE INTERPRETER REQUIRES MOST OF THE AVAILABLE PROCESSOR CYCLES TO DRIVE THE HIRES SCREEN (THE PRIMARY OBJECT). SO IT WOULD APPEAR THAT WE EITHER HAVE TO SETTLE FOR MONITORING THE INPUT PORT FREQUENTLY WHILE THE PICTURE SLOWS DOWN OR INFREQUENLY TO KEEP THE SCREEN MOVING.

THE DILEMMA PRESENTED ABOVE IS FURTHER COMPLICATED BY THE VARIATIONS IN THE TIME IT TAKES TO READ THE PORT AS REFLECTED BY THE SOUND PITCH AT ANY PARTICULAR SAMPLE. THAT IS, HIGHER NOTES TAKE LESS TIME TO READ THAN LOWER NOTES AND THE PACE OF EXECUTION WILL VARY DRAMATICALLY. UNLESS THE SCORE WAS DESIGNED TO TAKE ADVANTAGE OF THIS (WHICH IS POSSIBLE), THE VARIATIONS INTRODUCED MAY BE DISCONCERTING.

IN ADDITION, THERE ARE GOING TO BE SILENT PERIODS THAT HAVE TO BE HANDLED. CONSIDERING ALL THE ABOVE, A RATHER UNUSUAL APPROACH HAS BEEN DESIGNED WHICH IS NOW DESCRIBED. I HESITATE TO CALL IT A 'SOLUTION' BUT A COMPOSER MAY FIND IT SURPRISINGLY USEFUL AT TIMES.

A VARIABLE CALLED 'MUSESW' IS PROVIDED. IT CAN BE SET ON OR OFF BY A SCORE STATEMENT OR FROM THE KEYBOARD (SEE THE CHAPTERS ON EXECUTION COMMANDS AND VARIABLES). WHEN DETECTED 'ON', THE INTERPRETER WILL CONTINUOUSLY READ THE

INPUT PORT AND MAINTAIN A 'SOUND' STACK AND A 'QUIET' STACK OF EIGHT VARIABLES EACH. THESE ARE 'PUSH THRU' STACKS AND THE OLDEST SAMPLES SIMPLY 'FALL AWAY' AS THE LATEST ARE PUSHED ON.

WHEN A PERIOD OF CONTINUOUS SILENCE IS BROKEN, THE 'QUIET' STACK IS PUSHED DOWN AND THE TOP ENTRY SET TO ZERO. WHEN A PERIOD OF CONTINUOUS SOUND ENDS, THE 'SOUND' STACK IS UPDATED IN THE SAME MANNER. AT ANY GIVEN INSTANT EITHER 'SOUND' OR 'QUIET' WILL BE ZERO WHILE THE OTHER WILL BE NON-ZERO AND THE LOWER LEVELS OF BOTH STACKS WILL ALWAYS BE NON-ZERO (EXCEPT WHEN STARTING).

THIS TECHNIQUE PROVIDES A SHORT BUT ALWAYS CURRENT HISTORY OF THE ALTERNATION BETWEEN SOUND AND SILENCE AS THE INTERPRETER READS IT.

THE SOUND/QUIET VARIABLES ARE:

SOUND	QUIET
SOUND1	QUIET1
SOUND2	QUIET2
SOUND3	QUIET3
SOUND4	QUIET4
SOUND5	QUIET5
SOUND6	QUIET6
SOUND7	QUIET7

UNLIKE MOST MICROPROCESSOR STACKS, ALL OF THESE VARIABLES ARE DIRECTLY READABLE BY THE SCORE STATEMENTS.

AS MENTIONED PREVIOUSLY, AMPLITUDE CANNOT BE DIRECTLY READ FROM THE CASSETTE PORT. HOWEVER, ADJUSTMENTS TO THE VOLUME AND/OR FREQUENCY WILL CHANGE THE QUANTITY AND/OR QUALITY OF THE SOUND GETTING THRU TO THE INTERPRETER. OBVIOUSLY, THIS WILL ALSO EFFECT WHAT ONE IS HEARING. TO AVOID THIS PROBLEM, INTERPOSE (WHAT I CALL) A 'V-AMP' BEWTEEN THE SOUND SOURCE AND THE CASSETTE-IN PORT. ANY CHEAP MONORAL AMPLIFIER SHOULD BE ADEQUATE FOR THIS.

AS STATED BEFORE, TURNING THE 'MUSESW' ON WILL SLOW EXECUTION TO SOME EXTENT. THE INTERPRETER HOWEVER, DOES 'INTELLIGENT SAMPLING' TO MINIMIZE SPEED LOSS DURING EXTENDED PERIODS OF SILENCE.

STACKS

CEEMAC CONTAINS SEVERAL INTERNALLY SUPPORTED STACKS. TWO STACKS (SOUND & QUIET) ARE DEDICATED TO KEEPING THE LATEST INFORMATION WHEN READING THE CASSETTE-IN PORT FOR MUSIC.

A SEMI-INTELLIGENT STACK IS USED FOR THE 'PUSH' AND 'PULL' MACROS. THESE STACKS ARE ESSENTIALLY TRANSPARENT TO THE COMPOSER; THAT IS, HE NEEDN'T BE AWARE OF THEIR EXISTANCE AND THEY ARE SIMPLY MENTIONED HERE IN PASSING.

OF MORE IMMEDIATE INTEREST ARE THE 'AUTO-STAKS'. THERE IS ONE FOR BLINES AND ONE FOR SPLINES. WHENEVER THESE DRAWING MACROS ARE EXECUTED, THE INTERPRETER AUTOMATICLY 'PUSHES' THE PERTINENT INFORMATION ONTO THE APPROPRIATE STACK. THIS IS DONE TO SUPPORT THE 'BRASE' AND 'SPRASE' MACROS.

THE BLINE AUTO-STAK HAS A DEPTH OF 64 LINES. THE DEPTH LEVELS ARE NUMBERED 0-$3F (0-63) WHERE 0 IS THE LATEST BLINE DRAWN AND $3F IS THE OLDEST THAT CAN BE ERASED. AS A NEW BLINE IS DRAWN, ALL OTHERS ARE 'PUSHED THRU' WITH THE LAST ONE JUST FALLING AWAY. THE BRASE MACRO REQUIRES THIS 'AGE' BE SUPPLIED IN PARAM1. VALUES EXCEEDING $3F ARE 'MOD'ED $3F (THAT IS, $40 BECOMES $0, $41 BECOMES $1, ETC.).

THE SPLINE AUTO-STAK HAS A DEPTH OF 32 LINES. DEPTH LEVELS ARE 0-$1F. INVALID PARAM1 VALUES GIVEN WITH THE 'SPRASE' MACRO ARE 'MOD'ED $1F.

IT IS NOT NECESSARY (OR POSSIBLE) TO SPECIFY THE COLOR FOR ERASING AS IT WILL ALWAYS BE BLACK. A FUTURE CEEMAC RELEASE WILL ALLOW PARAM2 TO SPECIFY THE ERASE COLOR.

YOU SHOULD BE AWARE THAT THE 'BRASE AND 'SPRACE' COMMANDS JUST READ THE STACK ENTRY. ITEMS CAN DISAPPEAR FROM THE STACK ONLY BY BEING 'PUSHED' OUT THE BOTTOM.

THE INTENDED PURPOSE OF ALL THIS IS TO PERMIT THE COMPOSER TO PERFORM 'FOLLOWING ERASE' OPERATIONS ON THESE LINES. BY ALTERING THE 'AGE' OF THE LINE BEING ERASED, A SERIES OF LINES CAN TAKE ON THE APPEARANCE OF A PLANE OF VARYING WIDTH AND DENSITY AS IT MOVES ABOUT THE SCREEN. OTHER USES WILL, NO DOUBT, BE UNCOVERED.